Berechnung von Kontaktkräften für eine zwangsbasierte Dynamiksimulation

Thomas Warken

14. April 1999

Diplomarbeit nach einem Thema von Prof. Dr. Günter Hotz
in der Technischen Fakultät, Fachbereich 14 - Informatik,
der Universität des Saarlandes, Saarbrücken

Hiermit versichere ich an Eides Statt, daß ich diese Arbeit nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Saarbrücken, im März 1999

Inhaltsverzeichnis

1 Einleitung

Mit fortschreitender Technik wird der Bedarf an Simulationswerkzeugen immer größer. Produkte werden immer komplizierter und damit der Entwicklungsprozeß, angefangen beim Produktdesign bis hin zum Zusammenbau und Testen, immer Zeit- und Kostenintensiver. Das Design muß so gut durchdacht sein, daß der spätere Zusammenbau des Produktes nicht zu schwierig oder gar unmöglich wird. Hier sind Simulationswerkzeuge gefragt, mit deren Hilfe man schon in der Planungsphase den Montageprozeß simulieren, und so Fehler im Design frühzeitig erkennen und beheben kann. Auch wird es zunehmend wünschenswert, die Funktionalität des Produktes schon während der Planung zu simulieren, damit Designfehler nicht erst nach der Herstellung eines teuren Prototyps erkannt werden.

Die Wichtigkeit solcher Hilfsmittel ist zum Beispiel in der Automobilindustrie deutlich zu sehen, wo heute der Entwurf eines Produktes alleine mit Bleistift und Papier absolut undenkbar wäre.

Aber nicht nur für die Produktentwicklung, sondern auch für das Erlernen des Umgangs mit fertigen Produkten werden Simulationen immer wichtiger. So ist zum Beispiel bei der Pilotenausbildung die Benutzung von Flugsimulatoren nicht mehr wegzudenken.

Abhängig davon, was simuliert werden soll, ergeben sich verschiedene Ansprüche an Simulationswerkzeuge. So ist es zum Beispiel für die Simulation eines Zusammenpralls beim Automobilentwurf (virtueller Crash Test) nicht wichtig, daß ein Benutzer mit der virtuellen Umgebung interagieren kann. Eine Realzeitsimulation ist daher nicht notwendig. In einer ersten Phase können die zeitaufwändigen Berechnungen durchgeführt werden, die dann in einer zweiten Phase visualisiert werden. Soll dagegen ein Montageprozeß simuliert werden, ist es wünschenswert, daß ein Benutzer interaktiv in das Geschehen eingreifen kann. Dazu ist es zum einen notwendig, geeignete Interaktionsmöglichkeiten zu schaffen, und zum anderen müssen die verwendeten Algorithmen realzeitfähig sein. Eine Anforderung, die man immer an eine Simulation stellt, ist die physikalische Korrektheit. Man macht zwar häufig vereinfachende Annahmen, wie zum Beispiel die Nichtverformbarkeit der beteiligten Objekte, aber unter diesen Annahmen sollen sich die Objekte so verhalten, wie es reale Objekte in der realen Welt auch tun würden.

Bei der Simulation von bewegten starren Körpern versteht man unter physikalischer Korrektheit die korrekte Simulation der Dynamik. Dabei gibt es zwei verschiedene Vorgehensweisen. Die impulsbasierte Simulation wendet bei Kollisionen zwischen Objekten die Stoßgesetze an, um die resultierende Bewegung zu ermitteln. Die zwangsbasierte Simulation dagegen formuliert für alle Kontaktpunkte zwischen Objekten Zwangsbedingungen, aus deren Einhaltung die Bewegungsänderung resultiert. Die vorliegende Arbeit beschreibt drei Verfahren zur Berechnung von Kontaktkräften für eine zwangsbasierte Dynamiksimulation.

Gliederung und Resultate

Im *zweiten* Kapitel wird die Software Bibliothek SiLVIA beschrieben, im Rahmen deren Entwicklung diese Arbeit entstanden ist. SiLVIA stellt eine Reihe von realzeitfähigen Algorithmen bereit, mit deren Hilfe man virtuelle Umgebungen modellieren und visualisieren, mit diesen Umgebungen interagieren und die Dynamik der Objekte in diesen Umgebungen simulieren kann. Die in dieser Arbeit vorgestellten Verfahren zur Dynamiksimulation sind Bestandteil dieser Bibliothek.

Im *dritten* Kapitel wird zunächst die im folgenden verwendete Notation erklärt. Danach wird definiert, was wir unter starren Körpern verstehen. Dabei werden wir vereinbaren, daß wir uns in dieser Arbeit auf Polyeder beschränken, die durch konvexe Polygone begrenzt sind. Die Konvexität der Polygone ist keine Einschränkung, da man jedes nichtkonvexe Polygon triangulieren kann. Wir werden beschreiben, wie wir Objekte mit Hilfe starrer Transformationen beliebig verschieben und rotieren können, und somit jede Lage und Orientierung im Raum erreichen können. Dabei werden wir drei verschiedene Beschreibungen von Rotationen und deren Umrechnungen ineinander kennenlernen, nämlich Rotationsmatrizen, Quaternionen oder Euler Parameter sowie die Achse-Winkel Darstellung. Da wir für eine physikalisch korrekte Simulation die physikalischen Eigenschaften eines Körpers (Lage des Schwerpunktes, Masse, Trägheitsmatrix) kennen müssen, werden wir kurz erläutern, wie man die Berechnung dieser Größen mit Hilfe des Gaußschen Integralsatzes und des Integralsatzes von Green auf Linienintegrale zurückführen kann, die geschlossen lösbar sind. Nachdem wir dann definiert haben, was ein System starrer Körper (eine *Welt*) ist, werden wir schließlich die Bewegungsgleichungen für einen starren Körper herleiten. Diese sind ein Satz von Differentialgleichungen, die wir mit Hilfe des Eulerschemas diskretisieren werden.

Im *vierten* Kapitel befassen wir uns mit Kollisionserkennungsverfahren. Wir werden dabei zuerst aufzeigen, daß eine rein statische Kollisionserkennung nur kleine Schrittweiten zuläßt und daß sie uns nicht alle Informationen liefert, die wir für eine Dynamiksimulation benötigen. Dann werden wir zeigen, daß eine exakte dynamische Kollisionserkennung zu einem Nullstellenproblem für Polynome fünften Grades führt, und daß diese Art der Kollisionserkennung daher für Realzeitanwendungen zu ineffizient ist. Wir werden dann als Kompromißlösung die in SiLVIA implementierte Kollisionserkennung vorstellen. Diese erhält als Eingabe die Objektkonfigurationen zu zwei diskreten Zeitpunkten und interpoliert die Distanzen zwischen den zu testenden Objektteilen linear. Wir werden das Verfahren zur Kante-Kante-Kollisionserkennung genauer betrachten, da dieses im Rahmen dieser Arbeit entwickelt und implementiert wurde.

Im *fünften* Kapitel beschäftigen wir uns dann mit der Berechnung von Kontaktkräften. Dazu definieren wir zunächst einige Begriffe und geben in grober Form den in SilVIA implementierten Algorithmus zur zwangsbasierten Bewegungssimulation an. Dieser Algorithmus wird eine Routine verwenden, die zu einer gegebenen Kontaktsituation und gegebenen Bewegungsvorgaben die Kräfte berechnet, die die Objekte daran hindern, sich gegenseitig zu durchdringen. Diese Kontaktkräfte werden dann in die linearisierten Bewegungsgleichungen eingesetzt, um die resultierende Bewegung zu ermitteln. Wir werden dann drei Verfahren zur Berechnung der Kontaktkräfte vorstellen. Das erste Verfahren geht von der Annahme aus, daß die vorliegenden Kontakte bilateral sind und formuliert die Zwangsbedingungen als nichtlineares Gleichungssystem, das dann mit Hilfe des Newton-Verfahrens gelöst wird. Das zweite Verfahren faßt die Kontakte als unilaterale Kontakte auf und formuliert ein nichtlineares Komplementaritätsproblem (NCP). Dieses wird in ein nichtlineares Gleichungssystem umgewandelt, das dann mit einem leicht modifizierten Newton-Verfahren gelöst wird. Das dritte Verfahren schließlich ist eine Alternative zum zweiten Verfahren. Es faßt die Kontakte ebenfalls als unilaterale Kontakte auf und formuliert eine Folge von linearen Komplementaritätsproblemen (LCP), die jeweils mit dem Lemke-Algorithmus gelöst werden.

Im *sechsten* Kapitel schließlich zeigen wir einige Resultate in Form von Beispielszenarien, die mit SiLVIA simuliert wurden.

Danksagungen

Ich bedanke mich herzlich bei Herrn Prof. Dr. Günter Hotz für die Vergabe der Diplomarbeit und die interessante Themenstellung. Mein besonderer Dank gilt Herrn Dr. Elmar Schömer für die individuelle Betreuung. Allen Kommilitonen und Mitarbeitern an diesem Lehrstuhl danke ich für die fruchtbaren Diskussionen und Anregungen, sowie für ein hervorragendes Arbeitsklima.
Insbesondere aber danke ich meiner Mutter und Ihrem Lebensgefährten Herrn Horst Schilke, die mir ein Studium überhaupt erst ermöglichten.

2 Die Software Bibliothek SiLVIA

In diesem Kapitel wollen wir kurz das SiLVIA Projekt beschreiben, im Rahmen dessen diese Arbeit entstanden ist. Dieses Projekt entstand am Lehrstuhl von Professor Hotz an der Universität des Saarlandes und begann mit Arbeiten zum Thema impulsbasierte und zwangsbasierte Dynamiksimulation. Aus diesen Arbeiten entsprang die Idee, die Resultate in einer Software Bibliothek zusammenzufassen und um weitere Algorithmen zu ergänzen.

SiLVIA steht für *Simulation Library for Virtual Reality and Interactive Applications*. Es handelt sich dabei also um eine (C++-) Bibliothek, die Routinen zur Modellierung und Visualisierung von virtuellen Welten und zur interaktiven Manipulation und Dynamiksimulation von Objekten in einer solchen virtuellen Welt zur Verfügung stellt. Ein Anwender von SiLVIA soll also in die Lage versetzt werden, eigene Anwendungen zu schreiben, in denen ein Benutzer mit einer virtuellen Welt interagieren kann, ohne sich dabei selbst um die aufwändigen Routinen zur Objektmanipulation, Kollisionserkennung oder Dynamiksimulation kümmern zu müssen. Bei der Beschreibung von SiLVIA orientieren wir uns an Abbildung 2.1, die den Aufbau der Bibliothek schematisch beschreibt. Den Kern von SiLVIA bildet die ACIS®Datenstruktur zur Repräsentation der Objekte. Ein Grund für die Verwendung dieser Datenstruktur ist, daß viele CAD-Systeme auch einen ACIS®-Kern haben. Dadurch haben wir die Möglichkeit, mit solchen CAD-Systemen erstellte Objekte leicht in SiLVIA zu übernehmen. Ein weiterer Grund für die Verwendung von ACIS®ist, daß diese Datenstruktur die Möglichkeit bietet, auch Objekte mit gekrümmten Begrenzungsflächen und -Kanten zu repräsentieren. Wir haben uns im Moment zwar auf Polyeder beschränkt, aber dabei soll es in der Zukunft nicht bleiben. Jedes Objekt kann zwar beliebig gut durch Polyeder approximiert werden, aber je besser diese Approximationen sind, desto mehr Flächen haben die entsprechenden Polyeder und desto mehr Rechenzeit benötigen Algorithmen, die mit diesen Polyedern arbeiten.

Auf diesem Kern bauen zwei Datenstrukturen auf, die zu einer erheblichen Beschleunigung bei der Kollisionserkennung beitragen. Die *hierarchische Raumpartitionierung* wird benutzt, um die Anzahl der für eine Kollision in Frage kommenden Objektpaare schnell zu reduzieren. Dazu wird der Raum hierarchisch in Würfel zerlegt. Haben zwei Objekte keine Raumwürfel gemeinsam, können sie sich nicht überlappen. Ansonsten geht man zu einer feineren Partitionierung über. Man iteriert dies, bis man entweder eine Überlappung der Objekte ausschließen kann oder die letzte Hierarchiestufe erreicht hat. Ist letzteres der Fall, muß man die beiden Objekte direkt gegeneinander auf Kollision testen.

Die *Hüllkörperhierarchie* ist eine hierarchische Approximation eines Objektes durch

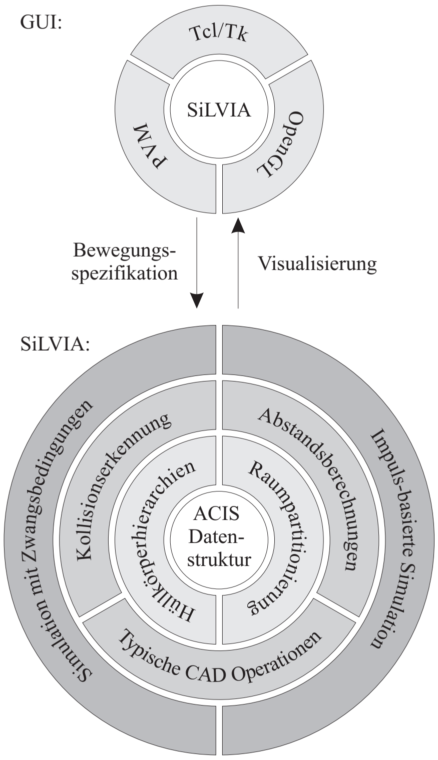

Abbildung 2.1: der Aufbau von SiLVIA

Hüllkörper. Dazu unterteilt man die Flächenmenge des Objektes in disjunkte Teilmengen, die man dann durch jeweils einen Hüllkörper (z.B. kleinste umschließende Kugeln) umschließt. Für diese Flächenmengen führt man dies rekursiv fort bis man entweder nur noch eine einzelne Fläche umhüllt oder keine geeignete Aufteilung mehr findet. Beim Kollisionstest zwischen zwei Objekten helfen die Hüllkörperhierarchien, die Anzahl der in Frage kommenden Flächenpaare zu reduzieren. Kollidieren die beiden Hüllkörper der gesamten Objekte nicht, können auch die Objekte selbst nicht kollidieren. Ansonsten testet man die Hüllkörper der nächsten Hierarchiestufe paarweise auf Kollision. Man iteriert dies, bis man entweder alle Flächen ausschließen kann oder die letzte Hierarchiestufe erreicht hat. Im zweiten Fall muß man die übriggebliebenen Flächen paarweise auf Kollision testen.

Zu den Algorithmen, die in SiLVIA implementiert sind, gehören zunächst einige *typische CAD-Operationen*. Dazu zählen Algorithmen zur Objektmanipulation (Verschieben, Rotieren), Routinen zum Erzeugen von Objekten (Extrusionskörper, Rotationskörper, boolesche Operationen auf Objekten) und Algorithmen zur Berechnung der physikalischen Momente (Schwerpunkte, Massen, Trägheitsmatrizen). Da zum einen die Kollisionserkennungsverfahren und die Verfahren zur Dynamiksimulation zur Zeit nur auf Polyedern arbeiten und zum anderen Polyeder schneller gerendert werden können, als

Objekte mit gekrümmten Begrenzungselementen, stellt SiLVIA auch einen Algorithmus zur Approximation von Objekten mit gekrümmten Begrenzungselementen durch Polyeder zur Verfügung.

Ein weiterer Algorithmus, der in der Bibliothek implementiert ist, ist ein Algorithmus zur *Abstandsberechnung*. Dieser liefert zu zwei Polyedern deren euklidische Distanz sowie zwei nächste Punkte auf den Polyedern. Für die Berechnung des Abstandes werden die Hüllkörperhierarchien benutzt, um eine branch-and-bound Strategie anzuwenden.

Der Algorithmus zur *Kollisionserkennung* benutzt wie schon erwähnt die Raumpartitionierung, um die Anzahl der Objektpaare schnell zu reduzieren und für die bleibenden Objektpaare die Hüllkörperhierarchien, um die Anzahl der Flächenpaare schnell zu verringern. Die elementaren Kollisionserkennungsroutinen benutzen die Tatsache, daß es bei Polyedern nur zwei Kollisionstypen gibt, nämlich die Ecke-Fläche- und die Kante-Kante-Kollision.

Schließlich gibt es noch Algorithmen zur Bewegungssimulation. Die *impulsbasierte Simulation* wendet bei einer Kollision von Objekten die Stoßgesetze an, um die resultierende Bewegung zu ermitteln. Dieser Algorithmus arbeitet zur Zeit noch nicht mit Mehrfachkontakten. Vielmehr werden Mehrfachkontakte durch eine Folge von einzelnen Kollisionen simuliert.

Die *Simulation mit Zwangsbedingungen* verwaltet eine Menge von Kontakten und formuliert für jeden Kontakt eine geometrische Zwangsbedingung. Aus diesen Zwangsbedingungen resultieren dann sogenannte Kontaktkräfte, aus denen sich die Bewegung der Objekte ergibt. Die Berechnung dieser Kontaktkräfte ist Gegenstand dieser Arbeit.

3 Grundlagen

In diesem Kapitel wollen wir die mathematischen und physikalischen Grundlagen für die später vorgestellten Verfahren zur Dynamiksimulation aufzeigen. Nachdem wir im ersten Abschnitt die in dieser Arbeit verwendete Notation erklärt haben haben, wollen wir definieren, was wir unter dem Begriff *starrer Körper* verstehen wollen. Nachdem wir eine Definition dieses Begriffs gegeben haben, beschäftigen wir uns mit Transformationen starrer Körper, beschreiben die für uns wichtigen physikalischen Eigenschaften und erklären schließlich, wie wir ein System starrer Körper beschreiben wollen. Der dritte Abschnitt beschäftigt sich dann schließlich mit der Bewegung von starren Körpern. Es werden die Bewegungsgleichungen in ihrer kontinuierlichen und diskretisierten Form hergeleitet.

3.1 Notation

Wir wollen an dieser Stelle die im folgenden verwendete Notation beschreiben und anschließend noch ein paar wenige Rechenregeln für Quaternionen angeben, die wir später benötigen werden.

- **Skalare** werden kursiv dargestellt. Beispielsweise sind i, n, K, α Skalare.
- **Matrizen** bezeichnen wir immer fett und aufrecht. Beispiele hierfür sind $\mathbf{M}, \mathbf{R}, \mathbf{I}$. Für die Einheitsmatrix schreiben wir immer $\mathbf{E}$ und für die Nullmatrix $\mathbf{0}$.
- **Vektoren** und vektorwertige Funktionen sind kursiv und fett dargestellt, wie zum Beispiel $\boldsymbol{a}, \boldsymbol{f}, \boldsymbol{F}, \boldsymbol{\omega}$. Wir fassen einen Vektor $\boldsymbol{a} \in \mathbb{R}^k$ immer als Spaltenvektor

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix}$$

auf. Der Nullvektor wird immer als $\mathbf{0}$ bezeichnet. Zu einem Vektor $\boldsymbol{a} = [a_1, a_2, a_3]^T \in \mathbb{R}^3$ bezeichnen wir mit $\boldsymbol{a}^\times$ die zu $\boldsymbol{a}$ gehörige schiefsymmetrische Matrix:

$$\boldsymbol{a}^\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}.$$

Für $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$ gilt damit $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{a}^\times \boldsymbol{b}$ und, wie man leicht nachrechnet,

$$\boldsymbol{a}^\times \boldsymbol{b}^\times = \boldsymbol{b}\boldsymbol{a}^T - \boldsymbol{a}^T\boldsymbol{b}\mathbf{E}. \tag{3.1}$$

- **Quaternionen** schreiben wir immer in fetter Fraktur, wie beispielsweise $\boldsymbol{\mathfrak{p}}, \boldsymbol{\mathfrak{q}}, \boldsymbol{\mathfrak{a}}$. Wir fassen Quaternionen als Spaltenvektoren auf. Ist

$$\boldsymbol{\mathfrak{q}} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix},$$

so bezeichnen wir mit $\boldsymbol{q} = [q_1, q_2, q_3]^T$ den Vektoranteil von $\boldsymbol{\mathfrak{q}}$ und schreiben auch

$$\boldsymbol{\mathfrak{q}} = \begin{pmatrix} q_0 \\ \boldsymbol{q} \end{pmatrix}.$$

Das zu $\boldsymbol{\mathfrak{q}}$ adjungierte Quaternion ist

$$\boldsymbol{\mathfrak{q}}^* = \begin{pmatrix} q_0 \\ -\boldsymbol{q} \end{pmatrix}.$$

Ist $\boldsymbol{a}$ ein Vektor, dann schreiben wir $\boldsymbol{\mathfrak{a}}$ für das Quaternion

$$\begin{pmatrix} 0 \\ \boldsymbol{a} \end{pmatrix}.$$

Wir schreiben dann manchmal auch $\boldsymbol{a}\boldsymbol{\mathfrak{q}}$ für das Quaternionenprodukt $\boldsymbol{\mathfrak{a}}\boldsymbol{\mathfrak{q}}$.

- **Mengen** schließlich bezeichnen wir immer mit großen Skript Buchstaben, wie $\mathcal{A}, \mathcal{O}, \mathcal{E}, \mathcal{K}$.

Wir werden später ein paar Rechenregeln verwenden, die wir jetzt angeben wollen. Zuerst geben wir die Definition des Quaternionenproduktes an. Seien $\boldsymbol{\mathfrak{p}}$ und $\boldsymbol{\mathfrak{q}}$ Quaternionen. Das Produkt $\boldsymbol{\mathfrak{p}}\boldsymbol{\mathfrak{q}}$ ist definiert als

$$\boldsymbol{\mathfrak{p}}\boldsymbol{\mathfrak{q}} = \begin{pmatrix} p_0 q_0 - \boldsymbol{p}^T\boldsymbol{q} \\ p_0\boldsymbol{q} + q_0\boldsymbol{p} + \boldsymbol{p} \times \boldsymbol{q} \end{pmatrix}. \tag{3.2}$$

Damit läßt sich für zwei Vektoren $\boldsymbol{a}$ und $\boldsymbol{b}$ das Kreuzprodukt $\boldsymbol{a} \times \boldsymbol{b}$ mit Hilfe des Quaternionenprodukts schreiben als

$$\begin{pmatrix} 0 \\ \boldsymbol{a} \times \boldsymbol{b} \end{pmatrix} = \frac{1}{2}(\boldsymbol{\mathfrak{a}}\boldsymbol{\mathfrak{b}} - \boldsymbol{\mathfrak{b}}\boldsymbol{\mathfrak{a}}). \tag{3.3}$$

Für ein Quaternion $\boldsymbol{\mathfrak{q}}$ gilt wegen (3.2) $|\boldsymbol{\mathfrak{q}}|^2 = \boldsymbol{\mathfrak{q}}\boldsymbol{\mathfrak{q}}^*$. Daher gilt für Quaternionen $\boldsymbol{\mathfrak{q}}$ mit $|\boldsymbol{\mathfrak{q}}| = 1$ die Gleichung $\boldsymbol{\mathfrak{q}}\boldsymbol{\mathfrak{q}}^* = 1$. Durch Differenzieren erhält man

$$\boldsymbol{\mathfrak{q}}\dot{\boldsymbol{\mathfrak{q}}}^* = -\dot{\boldsymbol{\mathfrak{q}}}\boldsymbol{\mathfrak{q}}^*. \tag{3.4}$$

Ferner gilt für Quaternionen $\mathfrak{p}$ und $\mathfrak{q}$

$$(\mathfrak{p}\mathfrak{q})^* = \mathfrak{q}^*\mathfrak{p}^*. \tag{3.5}$$

Schließlich werden wir noch die folgende Tatsache benötigen: Sei $\mathfrak{q}$ ein Quaternion. Gilt für jeden Vektor $\boldsymbol{a} \in \mathbb{R}^3$ die Gleichheit $\mathfrak{q}\mathfrak{a} = \mathfrak{a}\mathfrak{q}$, so gibt es ein Skalar s mit

$$\mathfrak{q} = \begin{pmatrix} s \\ \mathbf{0} \end{pmatrix}. \tag{3.6}$$

Beweis. Wegen (3.2) ist $\mathfrak{q}\mathfrak{a} = \mathfrak{a}\mathfrak{q}$ gleichbedeutend mit

$$\begin{pmatrix} -\boldsymbol{q}^T\boldsymbol{a} \\ q_0\boldsymbol{a} + \boldsymbol{q} \times \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{q}^T\boldsymbol{a} \\ q_0\boldsymbol{a} + \boldsymbol{a} \times \boldsymbol{q} \end{pmatrix}.$$

Dies ist äquivalent zu $\boldsymbol{q} \times \boldsymbol{a} = \boldsymbol{a} \times \boldsymbol{q}$. Da das Kreuzprodukt schiefsymmetrisch ist, folgt daraus $\boldsymbol{a} \times \boldsymbol{q} = \mathbf{0}$. Wenn dies für alle $\boldsymbol{a} \in \mathbb{R}^3$ gilt, muß $\boldsymbol{q} = \mathbf{0}$ sein. Damit ist $\mathfrak{q}$ von der Form $\mathfrak{q} = [s, \mathbf{0}^T]^T$ für ein Skalar s. □

3.2 Starre Körper

Unter starren Körpern verstehen wir Gegenstände, die nicht verformbar sind. Da in dieser Arbeit Verfahren vorgestellt werden sollen, die bei der Simulation der Bewegungen in einem System solcher Körper eingesetzt werden können (Kollisionserkennungsverfahren, Verfahren zur Berechnung sog. Kontaktkräfte), benötigen wir eine mathematische Repräsentation für starre Körper.

3.2.1 Definition

Wir bezeichnen eine Punktmenge $\mathcal{O} \subset \mathbb{R}^3$ als Körper, wenn sie folgende Eigenschaften erfüllt:

1. $\mathcal{O}$ ist kompakt.
2. $\mathcal{O}$ ist zusammenhängend.
3. $\mathcal{O} = \overline{int(\mathcal{O})}$,
   wobei $int(\mathcal{O})$ das Innere der Menge $\mathcal{O}$ und $\overline{\mathcal{A}}$ der Abschluß der Menge $\mathcal{A}$ ist.

Eigenschaft 3 besagt unter anderem, daß ein Körper nicht zu einem Flächen- oder Liniensegment oder zu einem Punkt entartet sein soll.

Definition 3.1 (starre Transformation). *Eine* starre Transformation *ist eine Abbildung* $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^3$ *mit* $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{R}\boldsymbol{x} + \boldsymbol{c}$*, wobei* $\boldsymbol{c} \in \mathbb{R}^3$ *und* $\mathbf{R} \in \mathbb{R}^{3\times 3}$ *mit*

1. $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{E}$,
2. $\det(\mathbf{R}) = 1$.

Wir bezeichnen einen Körper als starr, wenn als Transformationen für diesen Körper nur starre Transformationen erlaubt sind. Die Menge der starren Transformationen bildet bezüglich der Hintereinanderausführung eine Gruppe, die man die *eigentliche euklidische Bewegungsgruppe* nennt. Die Bezeichnung starr besagt, daß diese Transformationen längen- und winkelerhaltend sind. Die Matrix $\mathbf{R}$ nennen wir Rotationsmatrix und den Vektor $\boldsymbol{c}$ Translationsvektor. Ist $\boldsymbol{c} = \mathbf{0}$, so nennen wir die Transformation eine Drehung oder Rotation. Ist $\mathbf{R} = \mathbf{E}$, so nennen wir sie eine Verschiebung oder Translation.

Für unsere späteren Überlegungen ist das Innere des Objektes[1] nicht wichtig, so daß es für unsere Zwecke ausreicht, das Objekt durch seinen Rand $\partial\mathcal{O}$ zu beschreiben. Dabei machen wir noch eine weitere Einschränkung: $\mathcal{O}$ soll ein Polyeder sein. Dabei sollen die Polygone des Randes von $\mathcal{O}$ konvex sein, was keine Einschränkung darstellt, da man dies durch triangulieren immer erreichen kann.

3.2.2 Transformationen

Um Systeme starrer Körper aufzubauen, müssen wir in der Lage sein, Objekte beliebig im Raum zu plazieren und zu orientieren. Auch für die Beschreibung von Bewegungen müssen sich die Lage und die Orientierung eines Objektes verändern lassen. Das heißt, wir müssen Transformationen auf den Objekten durchführen können. Wie wir bereits gesehen haben, lassen wir als Transformationen nur die starren Transformationen zu. Wir müssen uns nun überlegen, daß diese Transformationen genügen, um jede mögliche Lage und Orientierung zu erhalten.

Definition 3.2. *Sei* $\mathcal{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ *eine Orthonormalbasis des* $\mathbb{R}^3$ *mit* $\det[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] = 1$*, und sei* $\boldsymbol{o} \in \mathbb{R}^3$*. Dann bezeichnen wir* $(\mathcal{E}, \boldsymbol{o})$ *als* Koordinatensystem mit Ursprung $\boldsymbol{o}$*. Ist* $\mathcal{E}$ *die Standardbasis und* $\boldsymbol{o} = \mathbf{0}$*, so heißt* $(\mathcal{E}, \boldsymbol{o})$ Weltkoordinatensystem*.*
Die Matrix $\mathbf{R} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3]$ *nennen wir* Orientierungsmatrix *oder* Orientierung *von* $(\mathcal{E}, \boldsymbol{o})$*.*

Wir bezeichnen also nur rechtsorientierte, orthogonale, kartesische Koordinatensysteme mit normierten Achsen als Koordinatensysteme in unserem Sinne.

Sei $\boldsymbol{a} \in \mathbb{R}^3$, und sei $(\mathcal{E}, \boldsymbol{o})$ ein Koordinatensystem mit Orientierung $\mathbf{R}$. Betrachten wir $\boldsymbol{a}$ als Punkt im Koordinatensystem $(\mathcal{E}, \boldsymbol{o})$, so repräsentiert $\boldsymbol{a}$ den Punkt mit den (Welt-) Koordinaten $\mathbf{R}\boldsymbol{a} + \boldsymbol{o}$.

Wir können nun ein Objekt $\mathcal{O}$ beliebig im Raum plazieren und orientieren, indem wir ein Koordinatensystem geeignet wählen, und jeden Punkt $\boldsymbol{a} \in \mathcal{O}$ bezüglich dieses Koordinatensystems interpretieren. Die Orientierungsmatrix dieses Systems erfüllt die Forderungen (1) und (2) in Definition 3.1, d.h. die Transformation in Weltkoordinaten ist eine starre Transformation.

Wir ordnen jedem Objekt $\mathcal{O}$ eine Rotationsmatrix $\mathbf{R}$ und einen Translationsvektor $\boldsymbol{o}$ zu, und bezeichnen das Koordinatensystem $(\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}, \boldsymbol{o})$ mit $\mathbf{R} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3]$ als

[1] Anstatt „starrer Körper“ sagen wir häufig auch „Objekt“.

Objektkoordinatensystem. Jeder Punkt $\hat{\boldsymbol{a}} \in \mathcal{O}$ wird dann im Objektkoordinatensystem interpretiert. Wir schreiben

$$\boldsymbol{a} = \mathbf{R}\hat{\boldsymbol{a}} + \boldsymbol{o} \tag{3.7}$$

um zu unterscheiden, in welchem Koordinatensystem wir uns befinden. $\hat{\boldsymbol{a}}$ ist immer der Punkt bzgl. des Objektkoordinatensystems und $\boldsymbol{a}$ bzgl. des Weltkoordinatensystems. Wir können die Rotationsmatrix $\mathbf{R}$ und den Translationsvektor $\boldsymbol{o}$ eines Objektes als Transformation des Weltkoordinatensystems in das Objektkoordinatensystem interpretieren: $\mathbf{R}$ beschreibt eine Rotation des Weltkoordinatensystems um eine Ursprungsachse und $\boldsymbol{o}$ eine Translation dieses rotierten Systems.

Wir wollen uns nun fragen, wie sich $\mathbf{R}$ und $\boldsymbol{o}$ ändern, wenn man ein Objekt um eine Achse durch den Punkt $\boldsymbol{p}$ in Richtung des Einheitsvektors $\boldsymbol{u}$ um den Winkel φ dreht. Wir bezeichnen die neue Rotationsmatrix mit $\mathbf{R}'$ und den neuen Translationsvektor mit $\boldsymbol{o}'$. Sei $\boldsymbol{a} = \mathbf{R}\hat{\boldsymbol{a}} + \boldsymbol{o}$ ein beliebiger Punkt des Objektes und sei $\boldsymbol{a}' = \mathbf{R}'\hat{\boldsymbol{a}} + \boldsymbol{o}'$ der Punkt nach der Rotation. Kennen wir die Matrix $\mathbf{R}_{\boldsymbol{u},\varphi}$, die eine Rotation um eine Ursprungsachse in Richtung $\boldsymbol{u}$ um den Winkel φ beschreibt, so können wir schreiben:

$$\begin{aligned}
\boldsymbol{a}' &= \mathbf{R}_{\boldsymbol{u},\varphi}(\boldsymbol{a} - \boldsymbol{p}) + \boldsymbol{p} \\
&= \mathbf{R}_{\boldsymbol{u},\varphi}(\mathbf{R}\hat{\boldsymbol{a}} + \boldsymbol{o} - \boldsymbol{p}) + \boldsymbol{p} = \mathbf{R}_{\boldsymbol{u},\varphi}\mathbf{R}\hat{\boldsymbol{a}} + \mathbf{R}_{\boldsymbol{u},\varphi}(\boldsymbol{o} - \boldsymbol{p}) + \boldsymbol{p}
\end{aligned}$$

Wegen $\boldsymbol{a}' = \mathbf{R}'\hat{\boldsymbol{a}} + \boldsymbol{o}'$ gilt somit

$$\begin{aligned}
\mathbf{R}' &= \mathbf{R}_{\boldsymbol{u},\varphi}\mathbf{R} \\
\boldsymbol{o}' &= \mathbf{R}_{\boldsymbol{u},\varphi}(\boldsymbol{o} - \boldsymbol{p}) + \boldsymbol{p}.
\end{aligned}$$

Wir müssen nun überlegen, wie $\mathbf{R}_{\boldsymbol{u},\varphi}$ aussieht. Sei dazu nun $\boldsymbol{a} \in \mathbb{R}^3$ ein beliebiger Punkt und sei $\boldsymbol{a}' = \mathbf{R}_{\boldsymbol{u},\varphi}\boldsymbol{a}$ der Punkt, der aus $\boldsymbol{a}$ durch die Rotation hervorgeht. Wie in Abbildung 3.1 dargestellt, liegen die Ortsvektoren von $\boldsymbol{a}$ und $\boldsymbol{a}'$ auf einem Kegel, dessen Achse durch den Einheitsvektor $\boldsymbol{u}$ gegeben ist. Aus Abbildung 3.1 wird ebenfalls ersichtlich, daß $\boldsymbol{a}' = \boldsymbol{a} + (1 - \cos\varphi)\boldsymbol{w} + \sin\varphi\boldsymbol{v}$ gilt. Dabei ist $\boldsymbol{v} = \boldsymbol{u} \times \boldsymbol{a}$ und $\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v}$. Dies können wir schreiben als

$$\begin{aligned}
\boldsymbol{a}' &= \boldsymbol{a} + (1 - \cos\varphi)\boldsymbol{u} \times (\boldsymbol{u} \times \boldsymbol{a}) + \sin\varphi(\boldsymbol{u} \times \boldsymbol{a}) \\
&= \mathbf{E}\boldsymbol{a} + (1 - \cos\varphi)\boldsymbol{u}^{\times}\boldsymbol{u}^{\times}\boldsymbol{a} + \sin\varphi\boldsymbol{u}^{\times}\boldsymbol{a} \\
&= \left(\mathbf{E} + (1 - \cos\varphi)\boldsymbol{u}^{\times}\boldsymbol{u}^{\times} + \sin\varphi\boldsymbol{u}^{\times}\right)\boldsymbol{a}.
\end{aligned}$$

Also ist die gesuchte Matrix

$$\begin{aligned}
\mathbf{R}_{\boldsymbol{u},\varphi} &= \mathbf{E} + (1 - \cos\varphi)\boldsymbol{u}^{\times}\boldsymbol{u}^{\times} + \sin\varphi\boldsymbol{u}^{\times} \\
&= \cos\varphi\mathbf{E} + (1 - \cos\varphi)\boldsymbol{u}\boldsymbol{u}^T + \sin\varphi\boldsymbol{u}^{\times}.
\end{aligned} \tag{3.8}$$

Dabei wurden (3.1) und $\boldsymbol{u}^T\boldsymbol{u} = 1$ benutzt. Gleichung (3.8) wird auch *Formel von Rodrigues* genannt.

Wir wollen noch eine alternative Schreibweise zu (3.8) herleiten, und zwar mit Hilfe der sog. *Euler Parameter*. Wir setzen $q_0 = \cos\frac{\varphi}{2}$ und $\boldsymbol{q} = \boldsymbol{u}\sin\frac{\varphi}{2}$. Schreiben wir

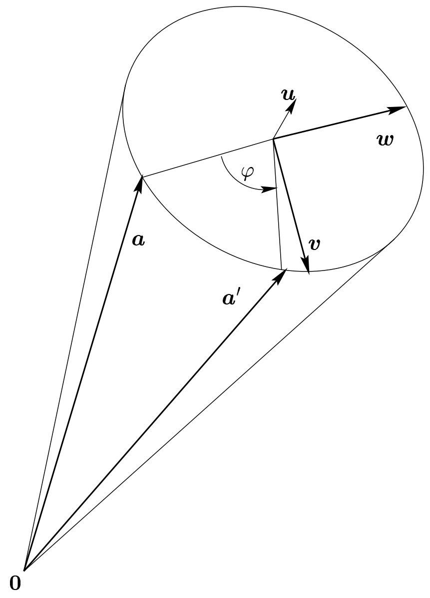

Abbildung 3.1: Rotation eines Punktes $\boldsymbol{a}$ um eine Ursprungsachse in Richtung des Einheitsvektors $\boldsymbol{u}$ um den Winkel φ.

$\boldsymbol{q} = (q_1, q_2, q_3)^T$, so bezeichnet man die vier Skalare q_0, q_1, q_2, q_3 als Euler Parameter. Man überlegt sich, daß folgende Beziehungen gelten:

$$\begin{array}{rclcl} \cos\varphi & = & \cos^2\frac{\varphi}{2} - \sin^2\frac{\varphi}{2} & = & q_0^2 - \boldsymbol{q}^T\boldsymbol{q} \\ \sin\varphi & = & 2\sin\frac{\varphi}{2}\cos\frac{\varphi}{2} & = & 2q_0|\boldsymbol{q}| \\ (1-\cos\varphi) & = & 2\sin^2\frac{\varphi}{2} & = & 2\boldsymbol{q}^T\boldsymbol{q} \end{array}$$

Damit wird aus (3.8)

$$\mathbf{R}_{\boldsymbol{u},\varphi} = (q_0^2 - \boldsymbol{q}^T\boldsymbol{q})\mathbf{E} + 2\boldsymbol{q}^T\boldsymbol{q}\boldsymbol{u}\boldsymbol{u}^T + 2q_0|\boldsymbol{q}|\boldsymbol{u}^\times.$$

Nun gelten aber

$$\begin{array}{rcl} \boldsymbol{q}^T\boldsymbol{q}\boldsymbol{u}\boldsymbol{u}^T & = & \sin^2\dfrac{\varphi}{2}\boldsymbol{u}\boldsymbol{u}^T \\ & = & (\sin\dfrac{\varphi}{2}\boldsymbol{u})(\sin\dfrac{\varphi}{2}\boldsymbol{u})^T \\ & = & \boldsymbol{q}\boldsymbol{q}^T \\ & \text{und} & \\ q_0|\boldsymbol{q}|\boldsymbol{u}^\times & = & q_0\sin\dfrac{\varphi}{2}\boldsymbol{u}^\times \\ & = & q_0\boldsymbol{q}^\times. \end{array}$$

Also erhalten wir

$$\mathbf{R}_{\boldsymbol{u},\varphi} = (q_0^2 - \boldsymbol{q}^T\boldsymbol{q})\mathbf{E} + 2\boldsymbol{q}\boldsymbol{q}^T + 2q_0\boldsymbol{q}^\times. \tag{3.9}$$

In Komponentenschreibweise sieht $\mathbf{R}_{\boldsymbol{u},\varphi}$ folgendermaßen aus:

$$\mathbf{R}_{\boldsymbol{u},\varphi} = \begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2+q_2^2-q_1^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2+q_3^2-q_1^2-q_2^2 \end{pmatrix}$$

An dieser Schreibweise sieht man, wie man bei gegebener Rotationsmatrix die Euler Parameter berechnen kann:

$$\begin{aligned} q_0^2 &= \frac{\mathrm{spur}(\mathbf{R}_{\boldsymbol{u},\varphi})+1}{4} \\ q_i^2 &= \frac{(\mathbf{R}_{\boldsymbol{u},\varphi})_{ii}+1}{2} - q_0^2 \qquad i=1,2,3 \end{aligned}$$

Dabei wurde $q_0^2+q_1^2+q_2^2+q_3^2=1$ benutzt. Die Vorzeichen der Parameter ergeben sich dann aus folgenden Überlegungen: Weil $q_0^2+\boldsymbol{q}^T\boldsymbol{q}=1$ gilt, muß mindestens eine Komponente von Null verschieden sein.

Fall 1: $q_0 \neq 0$
Wähle das Vorzeichen von q_0 beliebig. Dann ergeben sich q_1, q_2 und q_3 aus

$$\begin{aligned} (\mathbf{R}_{\boldsymbol{u},\varphi})_{21} - (\mathbf{R}_{\boldsymbol{u},\varphi})_{12} &= 4q_0q_3 \\ (\mathbf{R}_{\boldsymbol{u},\varphi})_{13} - (\mathbf{R}_{\boldsymbol{u},\varphi})_{31} &= 4q_0q_2 \\ (\mathbf{R}_{\boldsymbol{u},\varphi})_{32} - (\mathbf{R}_{\boldsymbol{u},\varphi})_{23} &= 4q_0q_1. \end{aligned}$$

Fall 2: $q_0 = 0,\ q_1 \neq 0$
Wähle das Vorzeichen von q_1 beliebig. Dann ergeben sich q_2 und q_3 aus

$$\begin{aligned} (\mathbf{R}_{\boldsymbol{u},\varphi})_{12} + (\mathbf{R}_{\boldsymbol{u},\varphi})_{21} &= 4q_1q_2 \\ (\mathbf{R}_{\boldsymbol{u},\varphi})_{13} + (\mathbf{R}_{\boldsymbol{u},\varphi})_{31} &= 4q_1q_3 \end{aligned}$$

Fall 3: $q_0 = q_1 = 0,\ q_2 \neq 0$
Wähle das Vorzeichen von q_2 beliebig. Dann ergibt sich q_3 aus

$$(\mathbf{R}_{\boldsymbol{u},\varphi})_{23} + (\mathbf{R}_{\boldsymbol{u},\varphi})_{32} = 4q_2q_3$$

Fall 4: $q_0 = q_1 = q_2 = 0,\ q_3 \neq 0$
Wähle das Vorzeichen von q_3 beliebig.

Euler Parameter haben gegenüber Darstellungen von Rotationen, die nur drei Komponenten haben (z.B. Euler Winkel), den Vorteil, daß man für die Transformation der Rotationsmatrix in diese Darstellung Formeln hat, deren rechte Seiten keine Singularitäten haben.

Schreiben wir $\mathfrak{q} = (q_0, q_1, q_2, q_3)^T$ und fassen die Euler Parameter als normiertes Quaternion auf, so rechnet man leicht nach, daß für ein beliebiges $\boldsymbol{a} \in \mathbb{R}^3$ gilt:

$$\begin{pmatrix} 0 \\ \mathbf{R}_{\boldsymbol{u},\varphi}\boldsymbol{a} \end{pmatrix} = \mathfrak{q}\mathfrak{a}\mathfrak{q}^* \tag{3.10}$$

Bestimmt man also zur Rotationsmatrix $\mathbf{R}$ eines Objektes $\mathcal{O}$ die Eulerparameter $\mathfrak{q}$, so gilt für jeden Punkt $\hat{\boldsymbol{a}} \in \mathcal{O}$

$$\mathfrak{a} = \mathfrak{q}\hat{\mathfrak{a}}\mathfrak{q}^* + \mathfrak{c}. \tag{3.11}$$

Wir nennen $\mathbf{q}$ das *Rotationsquaternion* zu Objekt $\mathcal{O}$. Wenn wir im folgenden von der *Konfiguration* eines Objektes reden, meinen wir damit den Ursprung und die Orientierung des Objektkoordinatensystems. Dabei unterscheiden wir nicht explizit, ob die Orientierung in Form eines Quaternions oder einer Matrix angegeben wird.

Wir haben nun drei Möglichkeiten gesehen, wie man eine Rotation beschreiben kann: Durch Angabe von Achse und Winkel, durch eine Rotationsmatrix und durch ein Rotationsquaternion. Wir haben auch gesehen, wie man aus der Achse-Winkel-Darstellung die Quaternionendarstellung und die Matrixdarstellung erhält. Ebenso haben wir gesehen, wie man aus der Matrixdarstellung die Quaternionendarstellung erhält. Um alle vorgestellten Darstellungen ineinander umrechnen zu können, müssen wir noch herleiten, wie man die Achse-Winkel-Darstellung aus einer der beiden anderen Darstellungen erhalten kann. Wir wollen dies für die Quaternionendarstellung tun. Sei $\mathbf{q} = (q_0, \boldsymbol{q}^T)^T$ ein normiertes Quaternion. Wir können oBdA davon ausgehen, daß $q_0 \geq 0$ ist. Ansonsten ersetzen wir q_0 durch $-q_0$ und $\boldsymbol{q}$ durch $-\boldsymbol{q}$. An Gleichung (3.9) sieht man, daß dadurch dieselbe Rotation beschrieben wird. Wir setzen dann einfach

$$\begin{aligned} \varphi &= 2 \arccos q_0 \\ \boldsymbol{u} &= \begin{cases} \frac{\boldsymbol{q}}{|\boldsymbol{q}|} & \boldsymbol{q} \neq \mathbf{0} \\ 0 & \boldsymbol{q} = \mathbf{0} \end{cases} \end{aligned} \tag{3.12}$$

Wie schon erwähnt, wollen wir davon ausgehen, daß unsere Objekte Polyeder sind, d.h. der Rand der Objekte besteht aus Ebenensegmenten. Daher wollen wir an dieser Stelle auch herleiten, wie man Ebenengleichungen transformiert. Sei also in einem Koordinatensystem mit Orientierungsmatrix $\mathbf{R}$ und Ursprung $\boldsymbol{o}$ eine Ebene gegeben, die der Gleichung $\hat{\boldsymbol{n}}^T \hat{\boldsymbol{x}} = \hat{n}_0$ genügt. Wir fragen uns also, wie diese Gleichung im Weltkoordinatensystem aussieht, d.h. wie man $\boldsymbol{n}$ und n_0 wählen muß, damit $\boldsymbol{n}^T \boldsymbol{x} = n_0$ gilt, wenn $\boldsymbol{x} = \mathbf{R}\hat{\boldsymbol{x}} + \boldsymbol{o}$ ist. Wegen $\hat{\boldsymbol{x}} = \mathbf{R}^T(\boldsymbol{x} - \boldsymbol{o})$ gilt $\hat{\boldsymbol{n}}^T \mathbf{R}^T (\boldsymbol{x} - \boldsymbol{o}) = \hat{n}_0$. Durch Umstellen erhalten wir daraus $(\mathbf{R}\hat{\boldsymbol{n}})^T \boldsymbol{x} = \hat{n}_0 + \boldsymbol{o}^T \mathbf{R}\hat{\boldsymbol{n}}$. Also wählen wir

$$\boldsymbol{n} = \mathbf{R}\hat{\boldsymbol{n}} \tag{3.13}$$

$$n_0 = \hat{n}_0 + \boldsymbol{o}^T \mathbf{R}\hat{\boldsymbol{n}} = \hat{n}_0 + \boldsymbol{o}^T \boldsymbol{n}. \tag{3.14}$$

3.2.3 Physikalische Eigenschaften

Für eine physikalische Simulation ist es natürlich unerläßlich, Masse, Schwerpunkt und Trägheitsmatrix jedes Objektes zu kennen. In diesem Abschnitt wollen wir nur kurz die Idee aufzeigen, wie man diese Größen bei Polyedern berechnen kann. Eine genaue Beschreibung des Algorithmus findet man in [Mir96a]

Die Definitionen der Masse m, des Schwerpunktes $\boldsymbol{c}$ und der Trägheitsmatrix $\mathbf{I}$ eines

Objektes $\mathcal{O}$ sind wie folgt:

$$m = \iiint_{\mathcal{O}} \rho d(x_1, x_2, x_3) \tag{3.15}$$

$$\boldsymbol{c} = \frac{1}{m} \iiint_{\mathcal{O}} \boldsymbol{x} \rho d(x_1, x_2, x_3) \tag{3.16}$$

$$\mathbf{I} = \iiint_{\mathcal{O}} (\boldsymbol{x}^2 \mathbf{E} - \boldsymbol{x}\boldsymbol{x}^T) \rho d(x_1, x_2, x_3) \tag{3.17}$$

Dabei ist $\boldsymbol{x} = (x_1, x_2, x_3)^T$ und $\rho = \rho(\boldsymbol{x})$ die Dichte im Punkt $\boldsymbol{x}$. Wir wollen annehmen, daß die Dichte auf dem Objekt konstant ist. Da wir uns auf Polyeder beschränken, können wir die Volumenintegrale mit Hilfe des Gaußschen Integralsatzes in Summen von Flächenintegralen verwandeln. Der Rand von Objekt $\mathcal{O}$ bestehe aus den n Polygonen $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Sei $\boldsymbol{p}_i \in \mathcal{A}_i$ für $i = 1, \ldots, n$, und $\mathcal{A}_i$ liege in der Ebene, die durch $\boldsymbol{p}_i + \alpha \boldsymbol{u}_i + \beta \boldsymbol{v}_i$ gegeben ist, wobei $\boldsymbol{u}_i^T \boldsymbol{v}_i = 0$ und $|\boldsymbol{u}_i| = |\boldsymbol{v}_i| = 1$. Definieren wir die Mengen $\mathcal{A}'_i = \{(\alpha, \beta)^T \in \mathbb{R}^2 | \boldsymbol{p}_i + \alpha \boldsymbol{u}_i + \beta \boldsymbol{v}_i \in \mathcal{A}_i\}$, dann können wir z.B. (3.15) schreiben als

$$m = \sum_{i=1}^{n} \iint_{\mathcal{A}'_i} \begin{pmatrix} \rho p_{i1} + \alpha \rho u_{i1} + \beta \rho v_{i1} \\ 0 \\ 0 \end{pmatrix}^T (\boldsymbol{u}_i \times \boldsymbol{v}_i) d(\alpha, \beta).$$

Zur besseren Übersicht setzen wir hierin

$$f\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (\rho p_{i1} + \alpha \rho u_{i1} + \beta \rho v_{i1})(\boldsymbol{u}_{i2} \boldsymbol{v}_{i3} - \boldsymbol{u}_{i3} \boldsymbol{v}_{i2})$$

und erhalten

$$m = \sum_{i=1}^{n} \iint_{\mathcal{A}'_i} f\begin{pmatrix} \alpha \\ \beta \end{pmatrix} d(\alpha, \beta). \tag{3.18}$$

Die Mengen $\mathcal{A}'_i$ sind wieder Polygone. Der Rand von $\mathcal{A}'_i$ bestehe aus den Liniensegmenten $\mathcal{S}_{i1}, \ldots, \mathcal{S}_{ik_i}$. Sei $\boldsymbol{q}_{ij} \in \mathcal{S}_{ij}$ für $i = 1, \ldots, n$ und $j = 1, \ldots, k_i$, und $\mathcal{S}_{ij}$ liege auf der Gerade, die durch $\boldsymbol{q}_{ij} + \gamma \boldsymbol{w}_{ij}$ gegeben ist, wobei $|\boldsymbol{w}_{ij}| = 1$. Definieren wir nun die Mengen $\mathcal{S}'_{ij} = \{\gamma \in \mathbb{R} | \boldsymbol{q}_{ij} + \gamma \boldsymbol{w}_{ij} \in \mathcal{S}_{ij}\}$, dann können wir (3.18) mit Hilfe des Integralsatzes von Green schreiben als

$$\begin{aligned} m &= \sum_{i=1}^{n} \sum_{j=1}^{k_i} \int_{\mathcal{S}'_{ij}} \begin{pmatrix} F(\boldsymbol{q}_{ij} + \gamma \boldsymbol{w}_{ij}) \\ 0 \end{pmatrix}^T \begin{pmatrix} w_{ij2} \\ -w_{ij1} \end{pmatrix} d\gamma \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k_i} \int_{\mathcal{S}'_{ij}} F(\boldsymbol{q}_{ij} + \gamma \boldsymbol{w}_{ij}) w_{ij2} d\gamma. \end{aligned} \tag{3.19}$$

Dabei haben wir

$$F\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \int_0^{\alpha} f\begin{pmatrix} a \\ \beta \end{pmatrix} da$$

gesetzt. Durch analoge Vorgehensweise kann man auch die Berechnung der Volumenintegrale (3.16) und (3.17) auf die Berechnung von Linienintegralen zurückführen. Alle diese Lininienintegrale sind geschlossen lösbar. Betrachten wir die Zeitdauer für die Berechnung einer solchen Lösung als konstant, so hat das Berechnen der physikalischen Momente lineare Komplexität in der Größe der Beschreibung des Objektes.

Wir wollen nun noch zwei Eigenschaften der Trägheitsmatrix herleiten, die wir später verwenden werden. Die erste ist fast offensichtlich, wenn man sich Gleichung (3.17) näher ansieht: $\mathbf{I}$ ist symmetrisch, d.h. $\mathbf{I} = \mathbf{I}^T$. Die zweite Eigenschaft besagt etwas über die Änderung der Trägheitsmatrix bei Transformation des Objektes. Gleichung (3.17) beschreibt die Trägheitsmatrix des Objektes $\mathcal{O}$ bezüglich des Ursprungs des Weltkoordinatensystems. Wollen wir die Trägheitsmatrix bezüglich des Schwerpunktes $\boldsymbol{c}$ berechnen, verschieben wir $\mathcal{O}$ so, daß $\boldsymbol{c} = \mathbf{0}$ wird und wenden dann (3.17) an. Ab jetzt bezeichnen wir mit $\mathbf{I}$ *immer* die Trägheitsmatrix bezüglich des Schwerpunktes. Wir wollen jetzt den Satz von Steiner beweisen, der besagt, wie man die Trägheitsmatrix $\mathbf{I}_0$ eines transformierten Objektes bezüglich des Ursprungs des Weltkoordinatensystems berechnen kann, wenn man die Trägheitsmatrix $\mathbf{I}_B$ des nicht transformierten Objektes bezüglich seines Schwerpunktes kennt. Wir definieren also

$$\begin{aligned}
\hat{\boldsymbol{c}} &= \frac{1}{m}\iiint_{\mathcal{O}} \hat{\boldsymbol{x}}\,dm \\
\mathbf{I}_B &= \iiint_{\mathcal{O}} (\hat{\boldsymbol{x}}^2\mathbf{E} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T)\,dm
\end{aligned}$$

und fordern, daß $\hat{\boldsymbol{c}} = \mathbf{0}$ ist. Man beachte, daß wir $\rho d(x_1, x_2, x_3) = dm$ schreiben können.

Satz 3.1 (Satz von Steiner). *Sei $\mathcal{O}$ ein Objekt mit Rotationsmatrix $\mathbf{R}$ und Translationsvektor $\boldsymbol{o}$. Dann gilt*

$$\mathbf{I}_0 = \mathbf{R}\mathbf{I}_B\mathbf{R}^T + m(\boldsymbol{o}^2\mathbf{E} - \boldsymbol{o}\boldsymbol{o}^T)$$

Beweis.

$$\begin{aligned}
\mathbf{I}_0 &= \iiint_{\mathcal{O}} (\boldsymbol{x}^2\mathbf{E} - \boldsymbol{x}\boldsymbol{x}^T)\,dm \\
&= \iiint_{\mathcal{O}} \left((\mathbf{R}\hat{\boldsymbol{x}} + \boldsymbol{o})^2\mathbf{E} - (\mathbf{R}\hat{\boldsymbol{x}} + \boldsymbol{o})(\mathbf{R}\hat{\boldsymbol{x}} + \boldsymbol{o})^T\right) dm \\
&= \iiint_{\mathcal{O}} (\hat{\boldsymbol{x}}^2\mathbf{E} - \mathbf{R}\hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\mathbf{R}^T)\,dm + \iiint_{\mathcal{O}} (\boldsymbol{o}^2\mathbf{E} - \boldsymbol{o}\boldsymbol{o}^T)\,dm \\
&\quad +2\iiint_{\mathcal{O}} \boldsymbol{o}^T\mathbf{R}\hat{\boldsymbol{x}}\mathbf{E}\,dm - \iiint_{\mathcal{O}} \mathbf{R}\hat{\boldsymbol{x}}\boldsymbol{o}^T\,dm - \iiint_{\mathcal{O}} \boldsymbol{o}\hat{\boldsymbol{x}}^T\mathbf{R}^T\,dm \\
&= \mathbf{R}\mathbf{I}_B\mathbf{R}^T + m(\boldsymbol{o}^2\mathbf{E} - \boldsymbol{o}\boldsymbol{o}^T) + 2\boldsymbol{o}^T\mathbf{R}m\hat{\boldsymbol{c}}\mathbf{E} - \mathbf{R}m\hat{\boldsymbol{c}}\boldsymbol{o}^T - \boldsymbol{o}(\mathbf{R}m\hat{\boldsymbol{c}})^T \\
&= \mathbf{R}\mathbf{I}_B\mathbf{R}^T + m(\boldsymbol{o}^2\mathbf{E} - \boldsymbol{o}\boldsymbol{o}^T)
\end{aligned}$$

□

Indem wir $\boldsymbol{o} = \mathbf{0}$ setzen, bekommen wir die Trägheitsmatrix $\mathbf{I}$ des transformierten Objektes bezüglich seines Schwerpunktes und es gilt

$$\mathbf{I} = \mathbf{R}\mathbf{I}_B\mathbf{R}^T \tag{3.20}$$

3.2.4 Systeme starrer Körper

Dieser Abschnitt soll erklären, was wir unter einem System von starren Körpern verstehen, das wir meistens als *Welt* bezeichnen. Sei $n \in \mathbb{N}$ und seien $\mathcal{O}_1, \ldots, \mathcal{O}_n$ starre Körper mit den Rotationsquaternionen $\mathbf{q}_1, \ldots, \mathbf{q}_n$ und den Schwerpunkten $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$. Dann bezeichnen wir die Menge $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ als *System starrer Körper* oder als *Welt*. Als Konfiguration einer Welt bezeichnen wir die Zusammenfassung der Konfigurationen der einzelnen Objekte. Diese schreiben wir als Vektor $\boldsymbol{s}$, der die folgende Form hat:

$$\boldsymbol{s} = [\boldsymbol{c}_1^T, \mathbf{q}_1^T, \ldots, \boldsymbol{c}_n^T, \mathbf{q}_n^T]^T \tag{3.21}$$

Seien $\mathbf{R}_1, \ldots, \mathbf{R}_n$ die zu den Rotationsquaternionen gehörigen Rotationsmatrizen. Wir bezeichnen die Konfiguration einer Welt als *legal*, falls für alle $1 \leq i < j \leq n$ gilt

$$(\mathbf{R}_i\mathcal{O}_i + \boldsymbol{c}_i) \cap (\mathbf{R}_j\mathcal{O}_j + \boldsymbol{c}_j) \subseteq \mathbf{R}_i\partial\mathcal{O}_i + \boldsymbol{c}_i, \tag{3.22}$$

wobei $\mathbf{R}\mathcal{O} + \boldsymbol{c} = \{\mathbf{R}\hat{\boldsymbol{a}} + \boldsymbol{c} | \hat{\boldsymbol{a}} \in \mathcal{O}\}$ ist. Diese Bedingung stellt sicher, daß keine Durchdringung zwischen zwei Objekten vorliegt. Wohl aber können Objekte sich berühren, d.h. die Schnittmenge der beiden Objekte muß Teil der Ränder dieser Objekte sein.

3.3 Bewegung

Um das dynamische Verhalten starrer Körper unter dem Einfluß von Kräften beschreiben zu können, wollen wir in diesem Abschnitt die Bewegungsgleichungen herleiten. Zu den Kräften, die auf die Objekte wirken, gehören zum Beispiel die Schwerkraft und die Kontaktkräfte, die wir in Kapitel 5 vorstellen werden.

Bewegung bedeutet, daß die Rotationsmatrix $\mathbf{R}$ und der Translationsvektor $\boldsymbol{o}$ eines Objektes zeitabhängig sind. Um einfachere Notationen zu erhalten, wollen wir vereinbaren, daß für den Schwerpunkt eines Objektes immer $\hat{\boldsymbol{c}} = \mathbf{0}$ gilt. Dies impliziert wegen Gleichung (3.7), daß $\boldsymbol{o} = \boldsymbol{c}$ gilt. Für einen Punkt $\boldsymbol{a}$ eines Objektes zum Zeitpunkt t in Weltkoordinaten können wir nun $\boldsymbol{a}(t) = \mathbf{R}(t)\hat{\boldsymbol{a}} + \boldsymbol{c}(t)$ schreiben. Für Objekte schreiben wir dann auch $\mathcal{O}(t) = \{\mathbf{R}(t)\hat{\boldsymbol{a}} + \boldsymbol{c}(t) | \hat{\boldsymbol{a}} \in \mathcal{O}\}$. Da wir später nur mit diskreten Zeitpunkten rechnen, werden wir meist das Zeitargument als oberen Index schreiben, um den Eindruck zu vermeiden, daß es sich um kontinuierliche Funktionen handelt. Wir schreiben also

$$\boldsymbol{a}^t = \mathbf{R}^t\hat{\boldsymbol{a}} + \boldsymbol{c}^t. \tag{3.23}$$

Ebenso ergibt sich aus den Gleichungen (3.13) und (3.14)

$$\boldsymbol{n}^t = \mathbf{R}^t\hat{\boldsymbol{n}} \quad \text{und} \tag{3.24}$$

$$n_0^t = \hat{n}_0 + (\boldsymbol{c}^t)^T\mathbf{R}^t\hat{\boldsymbol{n}} = \hat{n}_0 + (\boldsymbol{c}^t)^T\boldsymbol{n}^t. \tag{3.25}$$

Unter der *Geschwindigkeit* $\boldsymbol{v}$ eines Objektes versteht man die Bahngeschwindigkeit seines Schwerpunktes $\boldsymbol{c}$. Es gilt also $\boldsymbol{v} = \dot{\boldsymbol{c}}$. Natürlich interessiert uns auch die Zeitableitung der Rotationsmatrix $\mathbf{R}$. Nach Definition 3.1 gilt $\mathbf{R}\mathbf{R}^T = \mathbf{E}$ und daher $\frac{d}{dt}(\mathbf{R}\mathbf{R}^T) = \dot{\mathbf{R}}\mathbf{R}^T + \mathbf{R}\dot{\mathbf{R}}^T = \mathbf{0}$. Daraus folgt $\dot{\mathbf{R}}\mathbf{R}^T = -(\dot{\mathbf{R}}\mathbf{R}^T)^T$. Die Matrix $\dot{\mathbf{R}}\mathbf{R}^T$ ist also schiefsymmetrisch, d.h. es gibt einen (zeitabhängigen) Vektor $\boldsymbol{\omega}$ mit $\dot{\mathbf{R}}\mathbf{R}^T = \boldsymbol{\omega}^\times$. Nach $\dot{\mathbf{R}}$ aufgelöst haben wir

$$\dot{\mathbf{R}} = \boldsymbol{\omega}^\times \mathbf{R}. \tag{3.26}$$

Der Vektor $\boldsymbol{\omega}$ heißt die *Winkelgeschwindigkeit* der durch $\mathbf{R}$ beschriebenen Rotation. Wir wollen kurz erläutern, wie $\boldsymbol{\omega}$ zu interpretieren ist. Ein Objekt mit Schwerpunkt $\boldsymbol{c}$ habe die Rotationsmatrix $\mathbf{R}$ und die Winkelgeschwindigkeit sei $\boldsymbol{\omega}$. Für jeden Punkt $\boldsymbol{a}$ dieses Objektes gilt dann $\boldsymbol{a} = \mathbf{R}\hat{\boldsymbol{a}} + \boldsymbol{c}$. Die Schwerpunktsgeschwindigkeit sei Null, d.h. das Objekt führe eine reine Rotation um den Schwerpunkt aus. Die Geschwindigkeit dieses Punktes ist dann gegeben durch $\dot{\boldsymbol{a}} = \dot{\mathbf{R}}\hat{\boldsymbol{a}} = \boldsymbol{\omega}^\times \mathbf{R}\hat{\boldsymbol{a}} = \boldsymbol{\omega} \times (\boldsymbol{a} - \boldsymbol{c})$. Wählen wir für $\boldsymbol{a}$ einen Punkt mit $\boldsymbol{a} = \boldsymbol{c} + \lambda\boldsymbol{\omega}$ für ein $\lambda \in \mathbb{R}$, so ergibt sich nach dieser Überlegung für die Geschwindigkeit dieses Punktes $\dot{\boldsymbol{a}} = \mathbf{0}$. $\boldsymbol{\omega}$ ist also ein Vektor in Richtung der momentanen Drehachse. Berechnet man für jeden Zeitpunkt t wie in Abschnitt 3.2.2 gezeigt den zu $\mathbf{R}(t)$ gehörigen Winkel $\varphi(t)$, so kann gezeigt werden, daß $|\boldsymbol{\omega}| = \dot{\varphi}$ gilt. Diese Tatsache gibt der Winkelgeschwindigkeit ihren Namen.

3.3.1 Die kontinuierlichen Bewegungsgleichungen

Wenden wir uns nun den Bewegungsgleichungen zu. Als Bewegungsgleichungen bezeichnen wir die Gleichungen, die die Zeitableitungen der Schwerpunkte $\boldsymbol{c}$, der Orientierungsquaternionen $\mathbf{q}$, der Geschwindigkeiten $\boldsymbol{v}$ und der Winkelgeschwindigkeiten $\boldsymbol{\omega}$ beschreiben. Wir werden zur Berechnung von $\dot{\boldsymbol{\omega}}$ eine Formel für die Zeitableitung der inversen Trägheitsmatrix benötigen.

Lemma 3.1. *Für die Zeitableitung der inversen Trägheitsmatrix $\mathbf{I}^{-1}$ eines starren Körpers, der mit der Winkelgeschwindigkeit $\boldsymbol{\omega}$ um den Schwerpunkt rotiert, gilt*

$$\frac{d}{dt}\mathbf{I}^{-1} = \boldsymbol{\omega}^\times \mathbf{I}^{-1} - \mathbf{I}^{-1}\boldsymbol{\omega}^\times.$$

Beweis. Sei $\mathbf{R}$ die Rotationsmatrix des Objektes. Dann gilt wegen Gleichung (3.20) $\mathbf{I}^{-1} = \mathbf{R}\mathbf{I}_B^{-1}\mathbf{R}^T$. Differenzieren führt zu $\frac{d}{dt}\mathbf{I}^{-1} = \dot{\mathbf{R}}\mathbf{I}_B^{-1}\mathbf{R}^T + \mathbf{R}\mathbf{I}_B^{-1}\dot{\mathbf{R}}^T$. Wenden wir hierauf Gleichung (3.26) an und benutzen $(\boldsymbol{\omega}^\times)^T = -\boldsymbol{\omega}^\times$, so erhalten wir

$$\begin{aligned} \frac{d}{dt}\mathbf{I}^{-1} &= \boldsymbol{\omega}^\times \mathbf{R}\mathbf{I}_B^{-1}\mathbf{R}^T - \mathbf{R}\mathbf{I}_B^{-1}\mathbf{R}^T\boldsymbol{\omega}^\times \\ &= \boldsymbol{\omega}^\times \mathbf{I}^{-1} - \mathbf{I}^{-1}\boldsymbol{\omega}^\times. \end{aligned}$$

□

Satz 3.2 (die Bewegungsgleichungen). *Gegeben sei ein starrer Körper, auf den die Kräfte* $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_K$ *wirken. Ferner sei* $\boldsymbol{r}_k$ *der Vektor, der vom Schwerpunkt des Objektes zum Angriffspunkt der Kraft* $\boldsymbol{f}_k$ *zeigt. Dann gelten die folgenden Bewegungsgleichungen:*

$$\text{(i)} \qquad \dot{\boldsymbol{c}} = \boldsymbol{v} \tag{3.27}$$

$$\text{(ii)} \qquad \dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}\mathbf{q} \tag{3.28}$$

$$\text{(iii)} \qquad \dot{\boldsymbol{v}} = \frac{1}{m}\sum_{k=1}^{K} \boldsymbol{f}_k \tag{3.29}$$

$$\text{(iv)} \qquad \dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left(\sum_{k=1}^{K} \boldsymbol{r}_k \times \boldsymbol{f}_k - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}\right) \tag{3.30}$$

Beweis. Gleichung (i) ist gerade die Definition der Geschwindigkeit.
Gleichung (iii) geht direkt aus der dynamischen Grundgleichung hervor.
Zum Beweis von (ii) betrachten wir einen beliebigen Punkt $\boldsymbol{a}$ des Objektes. Dann gilt wegen (3.11) $\mathbf{a} = \mathbf{q}\hat{\mathbf{a}}\mathbf{q}^* + \mathbf{c}$. Daraus erhalten wir die beiden Gleichungen

$$\begin{aligned} \hat{\mathbf{a}}\mathbf{q}^* &= \mathbf{q}^*(\mathbf{a} - \mathbf{c}), \\ \mathbf{q}\hat{\mathbf{a}} &= (\mathbf{a} - \mathbf{c})\mathbf{q}. \end{aligned} \tag{3.31}$$

Ferner folgt durch Differenzieren $\dot{\mathbf{a}} = \dot{\mathbf{q}}\hat{\mathbf{a}}\mathbf{q}^* + \mathbf{q}\hat{\mathbf{a}}\dot{\mathbf{q}}^*$. Dabei wurde benutzt, daß bei einer Rotation um den Schwerpunkt $\dot{\boldsymbol{c}} = \mathbf{0}$ gilt. Mit (3.31) wird daraus $\dot{\mathbf{a}} = \dot{\mathbf{q}}\mathbf{q}^*(\mathbf{a} - \mathbf{c}) + (\mathbf{a} - \mathbf{c})\mathbf{q}\dot{\mathbf{q}}^*$. Andererseits gilt aber auch $\dot{\boldsymbol{a}} = \boldsymbol{\omega} \times (\boldsymbol{a} - \boldsymbol{c})$. Dies läßt sich nach (3.3) schreiben als $\dot{\mathbf{a}} = \frac{1}{2}(\boldsymbol{\omega}(\mathbf{a} - \mathbf{c}) - (\mathbf{a} - \mathbf{c})\boldsymbol{\omega})$. Damit haben wir $\dot{\mathbf{q}}\mathbf{q}^*(\mathbf{a} - \mathbf{c}) + (\mathbf{a} - \mathbf{c})\mathbf{q}\dot{\mathbf{q}}^* = \frac{1}{2}(\boldsymbol{\omega}(\mathbf{a} - \mathbf{c}) - (\mathbf{a} - \mathbf{c})\boldsymbol{\omega})$. Benutzt man (3.4), erhält man hieraus durch Umstellen

$$(\dot{\mathbf{q}} - \frac{1}{2}\boldsymbol{\omega}\mathbf{q})\mathbf{q}^*(\mathbf{a} - \mathbf{c}) = (\mathbf{a} - \mathbf{c})(\dot{\mathbf{q}} - \frac{1}{2}\boldsymbol{\omega}\mathbf{q})\mathbf{q}^*.$$

Wegen der beliebigen Wahl von $\boldsymbol{a}$ können wir (3.6) anwenden. Es existiert also ein Skalar s mit

$$(\dot{\mathbf{q}} - \frac{1}{2}\boldsymbol{\omega}\mathbf{q})\mathbf{q}^* = \begin{pmatrix} s \\ \mathbf{0} \end{pmatrix}.$$

Daraus folgt

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}\mathbf{q} + s\mathbf{q}. \tag{3.32}$$

Wir zeigen nun, daß $s = 0$ gilt. Zunächst folgt aus (3.32) mittels (3.5) $\dot{\mathbf{q}}^* = \frac{1}{2}\mathbf{q}^*\boldsymbol{\omega}^* + s\mathbf{q}^*$. Damit ergeben sich die beiden Gleichungen

$$\dot{\mathbf{q}}\mathbf{q}^* = \frac{1}{2}\boldsymbol{\omega} + s \quad \text{und} \quad \mathbf{q}\dot{\mathbf{q}}^* = \frac{1}{2}\boldsymbol{\omega}^* + s.$$

Durch Addition dieser beiden Gleichungen erhalten wir mit Hilfe von (3.4) $0 = \frac{1}{2}(\boldsymbol{\omega} + \boldsymbol{\omega}^*) + 2s = 2s$ und damit $s = 0$. Damit ist (ii) bewiesen.
Zum Beweis von (iv) betrachten wir den Drehimpuls $\boldsymbol{L} = \mathbf{I}\boldsymbol{\omega}$. Damit läßt sich die Winkelgeschwindigkeit schreiben als $\boldsymbol{\omega} = \mathbf{I}^{-1}\boldsymbol{L}$. Differenzieren wir dies, so ergibt sich

$$\dot{\boldsymbol{\omega}} = \frac{d}{dt}\mathbf{I}^{-1}\boldsymbol{L} + \mathbf{I}^{-1}\dot{\boldsymbol{L}} = \frac{d\mathbf{I}^{-1}}{dt}\mathbf{I}\boldsymbol{\omega} + \mathbf{I}^{-1}\dot{\boldsymbol{L}}. \tag{3.33}$$

Nach dem Drehimpulssatz gilt $\dot{\boldsymbol{L}} = \sum_{k=1}^{K} \boldsymbol{r}_k \times \boldsymbol{f}_k$. Setzen wir dies in (3.33) ein und wenden Lemma 3.1 an, so erhalten wir das gewünschte Resultat. □

3.3.2 Die diskretisierten Bewegungsgleichungen

Satz 3.2 liefert uns ein Differentialgleichungssystem, dessen Lösung exakt den zeitlichen Verlauf der Konfiguration eines Objektes beschreibt, wenn man die Anfangskonfiguration und die Funktionen $\boldsymbol{F}(t) = \sum_{k=1}^{K(t)} \boldsymbol{f}_k(t)$ und $\boldsymbol{D}(t) = \sum_{k=1}^{K(t)} \boldsymbol{r}_k(t) \times \boldsymbol{f}_k(t)$ kennt. Wir gehen aber im folgenden davon aus, daß wir $\boldsymbol{F}$ und $\boldsymbol{D}$ nicht kennen. Dies ist sinnvoll, da wir eine interaktive Manipulation der Objekte zulassen wollen, und wir über das Verhalten des Benutzers keine Vorhersagen machen können. Nehmen wir also an, daß wir $\boldsymbol{F}$ und $\boldsymbol{D}$ nur zu diskreten Zeitpunkten $t, t + \Delta t, \ldots$ kennen und daß diese Funktionen in jedem Intervall $[t, t + \Delta t)$ konstant sind. Wir können dies noch verschärfen, indem wir sogar fordern, daß die Funktionen $K, \boldsymbol{f}_k$ und $\boldsymbol{r}_k$ auf diesen Intervallen konstant sind.

Um nun für eine gegebene Konfiguration zum Zeitpunkt t und gegebene $\boldsymbol{f}_k$ und $\boldsymbol{r}_k$ die Konfiguration zum Zeitpunkt $t + \Delta t$ anzunähern, diskretisieren wir die Differentialgleichungen mit Hilfe des Eulerschemas. Dabei wenden wir auf die Gleichungen (3.29) und (3.30) Vorwärtsdifferentiation und auf die Gleichungen (3.27) und (3.28) Rückwärtsdifferentiation an. Wir erhalten somit die diskretisierten Varianten der Bewegungsgleichungen:

$$\boldsymbol{c}^{t+\Delta t} = \boldsymbol{c}^t + \Delta t \boldsymbol{v}^{t+\Delta t} \tag{3.34}$$

$$\mathfrak{q}^{t+\Delta t} = \mathfrak{q}^t + \frac{1}{2}\Delta t \boldsymbol{\omega}^{t+\Delta t} \cdot \mathfrak{q}^t \tag{3.35}$$

$$\boldsymbol{v}^{t+\Delta t} = \boldsymbol{v}^t + \Delta t \frac{1}{m}\sum_{k=1}^{K} \boldsymbol{f}_k^t \tag{3.36}$$

$$\boldsymbol{\omega}^{t+\Delta t} = \boldsymbol{\omega}^t + \Delta t (\mathbf{I}^t)^{-1} \left(\sum_{k=1}^{K} \boldsymbol{r}_k^t \times \boldsymbol{f}_k^t - \boldsymbol{\omega}^t \times \mathbf{I}^t \boldsymbol{\omega}^t \right) \tag{3.37}$$

Wir werden später auch die Matrizenschreibweise von Gleichung (3.35) verwenden:

$$\mathbf{R}^{t+\Delta t} = \mathbf{R}(\mathfrak{q}^{t+\Delta t}) = \mathbf{R}\begin{pmatrix} 1 \\ \frac{1}{2}\Delta t \boldsymbol{\omega}^{t+\Delta t} \end{pmatrix} \cdot \mathbf{R}(\mathfrak{q}^t) = \Delta\mathbf{R} \cdot \mathbf{R}^t \tag{3.38}$$

Durch die geschickte Anwendung von Vorwärts- und Rückwärtsdifferentiation sind wir nun in der Lage, Gleichung (3.36) in (3.34) und Gleichung (3.37) in (3.35) und (3.38) einzusetzen. Damit können wir Lage und Orientierung zum Zeitpunkt $t + \Delta t$ bestimmen, wenn wir die zum Zeitpunkt t auf das Objekt wirkenden Kräfte kennen.

4 Kollisionen und Kollisionserkennung

Wenn sich Objekte in einer Welt nach den im vorigen Kapitel vorgestellten Bewegungsgleichungen bewegen, kann es passieren, daß die Welt in eine nicht legale Konfiguration gerät, d.h. es können Durchdringungen zwischen Objekten auftreten. Den Vorgang des sich gegenseitig Durchdringens nennen wir eine Kollision zwischen den beteiligten Objekten. In diesem Kapitel wollen wir diesen Begriff etwas genauer betrachten und uns anschließend einige Verfahren zur Kollisionserkennung ansehen.

Sei $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ eine Welt, die zum Zeitpunkt t die legale Konfiguration $\boldsymbol{s}(t)$ hat. Jedes Objekt $\mathcal{O}_i$ habe eine momentane Geschwindigkeit $\boldsymbol{v}_i(t)$ und eine momentane Winkelgeschwindigkeit $\boldsymbol{\omega}_i(t)$. Der zeitliche Verlauf von $\boldsymbol{s}$ ergebe sich durch die kontinuierlichen Bewegungsgleichungen. Gibt es ein $t' > t$, so daß $\boldsymbol{s}(t')$ nicht legal ist, so liegt eine Kollision vor. Sei nun $t_{col} \geq t$ der Zeitpunkt, für den gilt:

- für alle $t' \in [t, t_{col}]$ ist $\boldsymbol{s}(t')$ legal, und
- für alle $t' > t_{col}$ existiert ein $\tau \in (t_{col}, t']$, so daß $\boldsymbol{s}(\tau)$ nicht legal ist.

Wir nennen t_{col} *Kollisionszeitpunkt*. Der Kollisionszeitpunkt ist also der späteste Zeitpunkt, zu dem sich die Objekte noch nicht durchdringen. Man beachte, daß wir nur dann von einer Kollision reden, wenn sich die Objekte durchdringen würden, da bei einer bloßen Berührung keine Wechselwirkung (Impulsübertragung) zwischen den Objekten stattfindet. Liegt eine Kollision vor, und ist t_{col} der Kollisionszeitpunkt, so gibt es Objekte $\mathcal{O}_i$ und $\mathcal{O}_j$ mit $i \neq j$, für die $\mathcal{O}_i(t_{col}) \cap \mathcal{O}_j(t_{col}) \neq \emptyset$ gilt. Da wir voraussetzen, daß die Objekte Polyeder sind, überlegt man sich, daß es stets einen Punkt $\boldsymbol{p} \in \mathcal{O}_i(t_{col}) \cap \mathcal{O}_j(t_{col})$ gibt,

- der eine Ecke eines Polygons von $\mathcal{O}_i(t_{col})$ oder $\mathcal{O}_j(t_{col})$ ist, oder
- der sowohl auf dem Rand eines Polygons von $\mathcal{O}_i(t_{col})$ als auch auf dem Rand eines Polygons von $\mathcal{O}_j(t_{col})$ liegt.

Im ersten Fall reden wir von einer *Ecke-Fläche-Kollision* und im zweiten Fall von einer *Kante-Kante-Kollision* (treten beide Fälle gleichzeitig ein, fassen wir die Kollision als Ecke-Fläche-Kollision auf).

Wenden wir uns nun den Kollisionserkennungsverfahren zu. Um festzustellen, ob eine Welt in eine nicht legale Konfiguration übergeht, müßte man prinzipiell alle Objektpaare $(\mathcal{O}_i, \mathcal{O}_j)$ mit $i < j$ gegeneinander auf Kollision testen. Bei großen Szenerien kann dies ein großer Aufwand sein. Heuristiken wie hierarchische Raumpartitionierungen können dann hilfreich sein, um die Anzahl der in Frage kommenden Objektpaare zu verringern. Hat man ein Objektpaar gegeben, so müßte man alle Flächen des einen Objektes gegen alle Flächen des zweiten Objektes testen. Auch dies bedeutet bei komplizierten Objekten einen großen Aufwand. Hüllkörperhierarchien sind hier eine gute Heuristik, um die Anzahl der in Frage kommenden Flächenpaare zu verringern. Diese Heuristiken sollen aber nicht Gegenstand dieser Arbeit sein. An dieser Stelle wollen wir nur einen genaueren Blick auf die elementare Kollisionserkennung werfen, wobei wir uns auch nicht um entartete Fälle kümmern wollen.

4.1 Statische Kollisionserkennung

Eine statische Kollisionserkennung entscheidet nur, ob eine bestimmte Konfiguration legal ist oder nicht. Dazu testet man für jede Kante des einen Objektes, ob sie eine Fläche des jeweils anderen Objektes durchstößt. Sei also eine Kante des einen Objektes gegeben durch ihre Endpunkte $\boldsymbol{a}$ und $\boldsymbol{b}$, und seien $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l$ die Eckpunkte einer Fläche des anderen Objektes. Die Fläche liege in der Ebene mit der Gleichung $\boldsymbol{n}^T\boldsymbol{x} = n_0$. Die Punkte $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l$ seien im Gegenuhrzeigersinn bzgl. der Normalen $\boldsymbol{n}$ angeordnet. Wir haben vereinbart, daß alle begrenzenden Flächen der Objekte konvex sind. Somit kann man einen einfachen Kollisionstest angeben. Die Kante hat einen nicht leeren Schnitt mit der Fläche genau dann, wenn eine der beiden folgenden Bedingungen eintritt.

1. $\boldsymbol{n}^T\boldsymbol{a} \geq n_0 \wedge \boldsymbol{n}^T\boldsymbol{b} \leq n_0 \wedge \forall 1 \leq i \leq l : \det[\boldsymbol{b} - \boldsymbol{a}, \boldsymbol{p}_{i+1} - \boldsymbol{p}_i, \boldsymbol{a} - \boldsymbol{p}_i] \geq 0$
2. $\boldsymbol{n}^T\boldsymbol{a} \leq n_0 \wedge \boldsymbol{n}^T\boldsymbol{b} \geq n_0 \wedge \forall 1 \leq i \leq l : \det[\boldsymbol{a} - \boldsymbol{b}, \boldsymbol{p}_{i+1} - \boldsymbol{p}_i, \boldsymbol{b} - \boldsymbol{p}_i] \geq 0$

Dabei haben wir $\boldsymbol{p}_{l+1} = \boldsymbol{p}_1$ gesetzt. Eine Herleitung dieser Bedingungen findet man in [Cha95].

Diese Art der Kollisionserkennung reicht für unsere Zwecke nicht aus. Wie bereits erwähnt, kennen wir die Konfigurationen der Welt nur zu diskreten Zeitpunkten. Betrachten wir eine Situation wie in Abbildung 4.1. Man sieht, daß die Konfigurationen zu den Zeitpunkten t und $t + \Delta t$ beide legal sind. Trotzdem kollidieren die Objekte, wenn man von einer geradlinigen Bewegung von Objekt $\mathcal{O}_1$ ausgeht. Diese Kollision kann von einem statischen Kollisionserkennungsverfahren nicht erkannt werden. Man könnte dieses Problem vernachlässigen, wenn man die Schrittweite Δt hinreichend klein wählen würde. Dann würde aber die Kollisionserkennung sehr häufig aufgerufen werden, was zu hohen Laufzeitkosten führen würde. Ein weiteres Problem ist, daß wir für eine zwangsbasierte Dynamiksimulation daran interessiert sind, welche Ecke bzw. Kante eines Objektes mit welcher Fläche bzw. Kante eines anderen Objektes zum Zeitpunkt t_{col} kollidiert. Wir müssen auch wissen, wo der Kollisionspunkt liegt, damit wir die in den Bewegungsgleichungen vorkommenden Vektoren $\boldsymbol{r}_k$ berechnen können. Wir können also für die Kollisionserkennung die Bewegungen der Objekte nicht vernachlässigen.

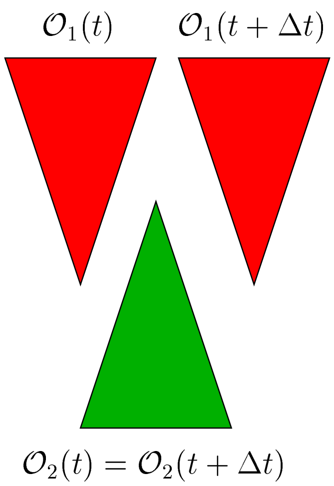

Abbildung 4.1: Obwohl zu allen diskreten Zeitpunkten die Konfiguration legal ist, können Kollisionen auftreten.

4.2 Exakte dynamische Kollisionserkennung

Wir wollen nun kurz ein Verfahren zur exakten dynamischen Kollisionserkennung betrachten (vgl. [Can87]). Dazu müssen wir zunächst einmal sagen, was wir unter „exakt“ verstehen wollen. Wir kennen die Objektkonfigurationen zu den Zeitpunkten t und $t + \Delta t$ und nehmen an, daß diese Konfigurationen sich gemäß der diskretisierten Bewegungsgleichungen ändern. Zum Zeitpunkt $t + \tau$ mit $\tau \in [0, \Delta t]$ seien die Schwerpunktskoordinaten und Orientierungsparameter also gegeben durch die Interpolationsgleichungen $\boldsymbol{c}_l(t+\tau) = \boldsymbol{c}_l^t + \tau \boldsymbol{v}_l^{t+\Delta t}$ und $\mathbf{q}_l(t+\tau) = \mathbf{q}_l^t + \frac{\tau}{2}\boldsymbol{\omega}_l^{t+\Delta t}\mathbf{q}_l^t$ für $l = 1, \ldots, n$. Die Geschwindigkeiten $\boldsymbol{v}_l^{t+\Delta t}$ und $\boldsymbol{\omega}_l^{t+\Delta t}$ seien dabei vorgegeben. Eine exakte dynamische Kollisionserkennung soll nun den gemäß dieser Bewegungsgleichungen exakten Kollisionszeitpunkt t_{col} finden (falls dieser existiert) und darüber hinaus ein Ecke-Fläche- oder Kante-Kante-Paar liefern, das zu diesem Zeitpunkt kollidiert, sowie den exakten Kollisionspunkt bestimmen. Dazu betrachten wir alle für eine Kollision in Frage kommenden Objektpaare $\mathcal{O}_i, \mathcal{O}_j$ separat (d.h. wir betrachten eine Welt, die nur aus diesen beiden Objekten besteht) und wählen dann das Minimum aller gefundenen Kollisionszeitpunkte mit dem zugehörigen Ecke-Fläche- oder Kante-Kante-Paar. Für ein gegebenes Objektpaar betrachten wir wieder alle in Frage kommenden Ecke-Fläche- und Kante-Kante-Paare separat. Wir müssen uns nun überlegen, wie wir für ein gegebenes Ecke-Fläche- bzw. Kante-Kante-Paar einen exakten dynamischen Kollisionstest durchführen. Wir wollen dies nur für den Ecke-Fläche-Fall tun, da die Vorgehensweise im Kante-Kante-Fall ähnlich ist, aber die Formeln komplizierter werden. Um herauszufinden, ob eine Ecke $\boldsymbol{a}$ von $\mathcal{O}_i$ mit einer Fläche von $\mathcal{O}_j$ kollidiert, die in der Ebene mit der Gleichung $\boldsymbol{n}^T\boldsymbol{x} - n_0 = 0$ liegt, müssen wir

1. testen, ob $\boldsymbol{a}$ während der Bewegung diese Ebene durchstößt. Falls ja, müssen wir

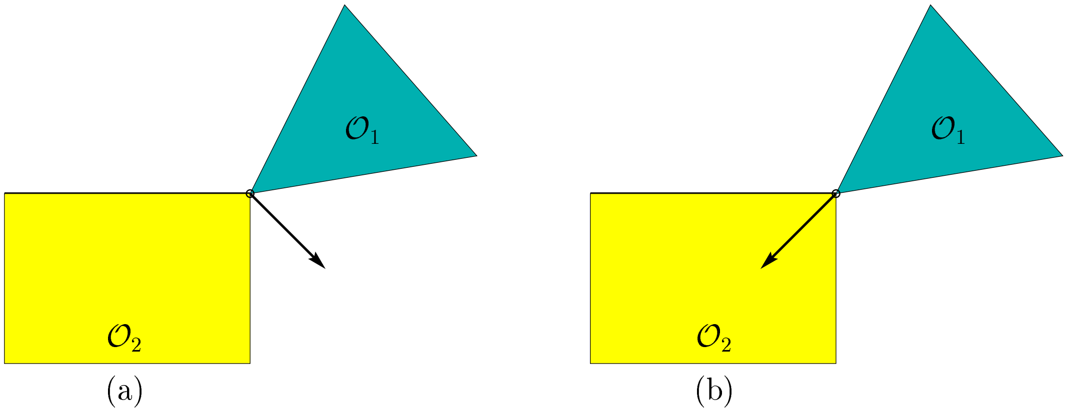

Abbildung 4.2: Beispiel für einen entarteten Fall beim Ecke-Fläche-Kollisionstest: Der Vektor gibt jeweils die Bewegungsrichtung der Ecke von $\mathcal{O}_1$ relativ zu $\mathcal{O}_2$ an. In Fall (a) können sich die Objekte aneinander vorbei bewegen, in Fall (b) kommt es zu einer Kollision

den Zeitpunkt t_{col} des Durchstoßens berechnen und

2. testen, ob der Durchstoßpunkt innerhalb der Fläche liegt.

Für den zweiten Test können wir wie bei der statischen Kollisionserkennung ausnutzen, daß die Fläche laut Annahme konvex ist. Der Durchstoßpunkt ist $\boldsymbol{a}(t_{col})$. Seien $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l, \boldsymbol{p}_{l+1} = \boldsymbol{p}_1$ die Eckpunkte der Fläche im Gegenuhrzeigersinn bzgl. der Normalenrichtung. Da $\boldsymbol{a}(t_{col})$ in der selben Ebene liegt wie die Fläche, befindet er sich genau dann innerhalb der Fläche, wenn er in dem durch die Fläche und ihren Normalenvektor definierten Extrusionskörper liegt, d.h. wenn gilt

$$\forall 1 \leq i \leq l : \det[\boldsymbol{n}(t_{col}), \boldsymbol{p}_{i+1}(t_{col}) - \boldsymbol{p}_i(t_{col}), \boldsymbol{a}(t_{col}) - \boldsymbol{p}_i(t_{col})] \geq 0. \tag{4.1}$$

Entscheiden wir bei diesem Test, daß $\boldsymbol{a}(t_{col})$ innerhalb der Fläche liegt, und war für ein (oder zwei) i die Determinante exakt gleich Null, so liegt ein entarteter Fall vor. Der Durchstoßpunkt liegt dann genau auf einer Kante (oder gar einer Ecke). In diesem Fall müssen wir die zur Ecke $\boldsymbol{a}$ adjazenten Flächen und die relative Bewegungsrichtung der Objekte betrachten, um sicher entscheiden zu können, ob eine Kollision vorliegt oder nicht (siehe Abbildung 4.2). Eine Behandlung dieses Spezialfalles findet der Leser in [Jos97].

Schwieriger wird der erste Test. Dazu definieren wir die Funktion

$$f(\tau) = \boldsymbol{n}(t+\tau)^T \boldsymbol{a}(t+\tau) - n_0(t+\tau).$$

Diese berechnet die vorzeichenbehaftete euklidische Distanz zum Zeitpunkt $t+\tau$ zwischen $\boldsymbol{a}$ und der Ebene, in der die Fläche liegt. Die Ecke durchstößt die Ebene genau dann von außen, wenn f eine Nullstelle mit Vorzeichenwechsel von positiv nach negativ hat.

Gesucht ist also die kleinste nicht negative Nullstelle von f mit dieser Eigenschaft, falls diese existiert. Dabei können wir

$$\begin{aligned}
\boldsymbol{a}(t+\tau) &= \mathbf{R}_i(t+\tau)\hat{\boldsymbol{a}} + \boldsymbol{c}_i(t+\tau) \\
\boldsymbol{n}(t+\tau) &= \mathbf{R}_j(t+\tau)\hat{\boldsymbol{n}} \\
n_0(t+\tau) &= \hat{n}_0 + (\boldsymbol{c}_j(t+\tau))^T \mathbf{R}_j(t+\tau)\hat{\boldsymbol{n}}
\end{aligned}$$

einsetzen. Wir würden nun gerne zur Quaternionenschreibweise übergehen und die Beziehung (3.10) benutzen. Diese Beziehung gilt aber nur für normierte Quaternionen. Diese Eigenschaft geht bei der Interpolation verloren. Durch Normieren des Quaternions wird aus (3.10)

$$\begin{pmatrix} 0 \\ \mathbf{R}_{\boldsymbol{u},\varphi}\boldsymbol{a} \end{pmatrix} = \frac{\mathfrak{q}\mathfrak{a}\mathfrak{q}^*}{\mathfrak{q}\mathfrak{q}^*}.$$

Setzen wir dies in unsere Funktion f ein und benutzen die Interpolationsgleichungen für $\boldsymbol{c}(t+\tau)$ und $\mathfrak{q}(t+\tau)$, so erhalten wir eine Funktion folgenden Aussehens:

$$f(\tau) = p_1(\tau) + \frac{p_2(\tau)}{p_3(\tau)}.$$

Dabei sind p_1, p_2, p_3 Polynome mit $grad(p_1) = 3$, $grad(p_2) = 2$ und $grad(p_3) = 2$. Zur Bestimmung der Nullstellen von f muß man also die Nullstellen eines Polynoms fünften Grades bestimmen. Falls f keine nicht negative Nullstelle mit den gewünschten Eigenschaften hat, kollidiert $\boldsymbol{a}$ nicht mit der Fläche. Ansonsten sei τ die kleinste solche Nullstelle. Dann ist $t_{col} = t + \tau$ der Kollisionszeitpunkt und $\boldsymbol{a}(t+\tau)$ der gesuchte Durchstoßpunkt, und der zweite Test kann durchgeführt werden.

Der Vollständigkeit halber sei erwähnt, daß man bei analoger Vorgehensweise im Kante-Kante-Fall ebenfalls die Nullstellen eines Polynoms fünften Grades bestimmen muß.

Die gerade geschilderte Vorgehensweise hat den Vorteil, exakt zu sein. Wenn man den exakten Kollisionszeitpunkt kennt, kann man mit Hilfe der Interpolationsgleichungen die Objekte exakt in die Lage bringen, in der sie zum Zeitpunkt der Kollision sein sollten. Nachteilhaft hingegen ist, daß man Nullstellen eines Polynoms fünften Grades berechnen muß. Darüber hinaus sind die Koeffizienten dieses Polynoms sehr komplizierte Terme. Alles in allem ist ein solcher exakter Kollisionstest sehr teuer.

4.3 Ein Kompromiß

Wir haben gesehen, daß sowohl eine rein statische als auch eine exakte dynamische Kollisionserkennung Nachteile haben, die wir nicht akzeptieren wollen. Die statische Kollisionserkennung liefert nicht alle Informationen, an denen wir interessiert sind und läßt nur sehr kleine Werte für Δt zu. Die exakte dynamische Kollisionserkennung liefert zwar alle Informationen, die wir brauchen und läßt auch große Werte für Δt zu, ist aber zu ineffizient, wenn man bedenkt, daß diese Routine im worst-case quadratisch oft in der Größe der Objektbeschreibungen aufgerufen wird, und daß jedesmal die Nullstellen eines komplizierten Polynomes fünften Grades bestimmt werden müssen. Wir wollen daher einen Kompromiß eingehen. Das bedeutet, daß wir ein Verfahren haben wollen, das effizient ist, das in der Lage ist, den Kollisionszeitpunkt und die an der Kollision beteiligten Objektteile zu bestimmen und das für vernünftig große Zeitschritte noch ausreichend exakt ist. Da das Eingehen eines Kompromisses einen Genauigkeitsverlust bedeutet, müssen wir damit rechnen, daß für zu große Δt ähnliche Effekte auftreten können, wie in Abbildung 4.1 gezeigt. Daher nehmen wir ab jetzt immer an, daß Δt hinreichend klein gewählt ist.

Wir wollen nun ein Verfahren sowohl für den Ecke-Fläche-Test als auch für den Kante-Kante-Test angeben. Dabei wollen wir auf die Behandlung von Spezialfällen wie Ecke-Kante- und Ecke-Ecke-Kollision nicht näher eingehen, sondern den Leser hierfür auf [Jos97] verweisen.

Ecke-Fläche-Test

Sei wie zuvor $\boldsymbol{a}$ eine Ecke des einen Objektes. Die Fläche des anderen Objektes, die wir mit $\boldsymbol{a}$ auf Kollision testen wollen, liege wieder in der Ebene mit der Gleichung $\boldsymbol{n}^T\boldsymbol{x} = n_0$. Die Objektkonfigurationen und damit die Werte von $\boldsymbol{a}, \boldsymbol{n}$ und n_0 seien zu den Zeitpunkten t und $t + \Delta t$ bekannt. Wir wollen nun nicht mehr wie bei obigem exaktem Test die diskretisierten Bewegungsgleichungen benutzen (d.h. die Konfigurationen linear interpolieren), sondern die Distanzen zu den Zeitpunkten t und $t+\Delta t$ berechnen und diese linear interpolieren. Wie wir gesehen haben, ist die Distanzfunktion bei linearer Interpolation der Objektkonfigurationen ein Polynom fünften Grades. Daher ist es klar, daß wir bei der Linearisierung, die wir jetzt vornehmen, ein hinreichend kleines Δt benutzen müssen. Wir definieren

$$\begin{aligned} d^t &= (\boldsymbol{n}^t)^T \boldsymbol{a}^t - n_0^t \qquad \text{und} \\ d^{t+\Delta t} &= (\boldsymbol{n}^{t+\Delta t})^T \boldsymbol{a}^{t+\Delta t} - n_0^{t+\Delta t} \end{aligned}$$

und setzen

$$d(\tau) = d^t + \frac{\tau}{\Delta t}(d^{t+\Delta t} - d^t).$$

Wir entscheiden nun folgendermaßen, ob $\boldsymbol{a}$ mit der Ebene kollidiert (siehe Abbildung 4.3):

- Falls $d(0) < -\varepsilon$ oder $d(0) \leq d(\Delta t)$
  $\Longrightarrow$ keine Kollision.
- Ansonsten kollidiert $\boldsymbol{a}$ mit der Ebene zum Zeitpunkt

$$t_c = \begin{cases} 0 & \text{falls } d(0) \leq 0 \\ \Delta t \frac{d(0)}{d(0)-d(\Delta t)} & \text{falls } d(0) > 0. \end{cases}$$

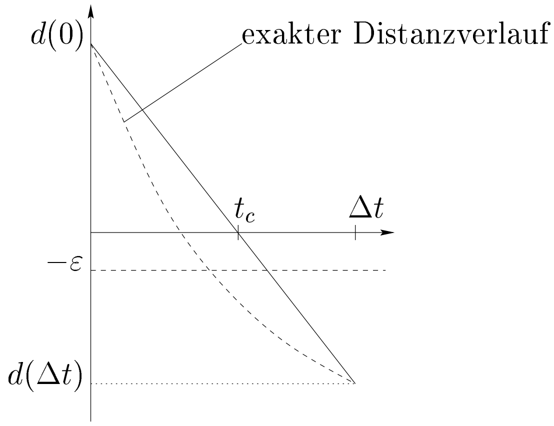

Abbildung 4.3: Lineare Interpolation der Distanzfunktion

Dabei ist $\varepsilon > 0$ die *Durchdringungstoleranz.* Diese soll numerische Ungenauigkeiten ausgleichen, die im Laufe der Dynamiksimulation auftreten können. Dieser Test entscheidet, daß keine Kollision vorliegt, falls $\boldsymbol{a}$ zum Zeitpunkt t auf der falschen Seite der Ebene liegt oder falls $\boldsymbol{a}$ sich von der Ebene entfernt. Ansonsten wird der Zeitpunkt $t + t_c$, zu dem $\boldsymbol{a}$ die Ebene durchstößt, durch lineare Interpolation ermittelt. Um dabei zu verhindern, daß wir einen negativen Kollisionszeitpunkt erhalten, wird t_c auf Null gesetzt, wenn die Distanz zum Zeitpunkt t negativ ist.

Nachdem wir festgestellt haben, daß die Ecke die Ebene durchstößt, müssen wir wie bei dem exakten Verfahren prüfen, ob der Durchstoßpunkt innerhalb oder außerhalb der Fläche liegt. Seien dazu wieder $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l, \boldsymbol{p}_{l+1} = \boldsymbol{p}_1$ die Eckpunkte der Fläche im Gegenuhrzeigersinn um die Normalenrichtung. Hätten wir t_c exakt bestimmt, so könnten wir mit Hilfe der diskretisierten Bewegungsgleichungen die exakten Objektkonfigurationen zum Zeitpunkt $t + t_c$ bestimmen und Kriterium (4.1) benutzen. Da t_c nicht exakt ist, ist es nicht weiter schlimm, auch die in (4.1) vorkommenden Determinanten zu interpolieren. D.h. wir definieren

$$\sigma_i^t = \det[\boldsymbol{n}^t, \boldsymbol{p}_{i+1}^t - \boldsymbol{p}_i^t, \boldsymbol{a}^t - \boldsymbol{p}_i^t]$$

und analog $\sigma_i^{t+\Delta t}$. Wir interpolieren σ_i linear:

$$\sigma_i(\tau) = \sigma_i^t + \frac{\tau}{\Delta t}(\sigma_i^{t+\Delta t} - \sigma_i^t)$$

Unser Entscheidungskriterium lautet dann

$$\forall 1 \leq i \leq l : \sigma_i(t_c) \geq 0.$$

Man beachte, daß dieser Test auch damit zurecht kommt, daß $\boldsymbol{a}$ zum Zeitpunkt $t + t_c$ aufgrund von Interpolationsfehlern nicht exakt in der Ebene liegt. Das liegt daran, daß der Test für alle Punkte positiv ausfällt, die in dem durch die Fläche und den Normalenvektor $\boldsymbol{n}$ definierten Extrusionskörper liegen.

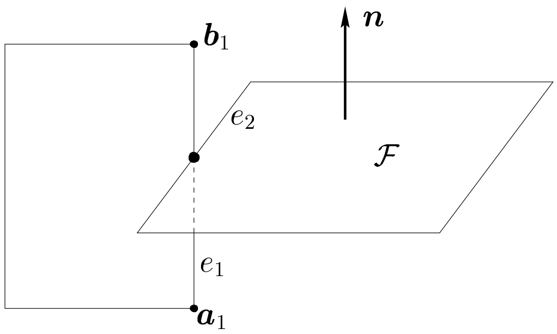

Abbildung 4.4: Die Kanten e_1 und e_2 können miteinander kollidieren, weil sich $\boldsymbol{a}_1$ und $\boldsymbol{b}_1$ auf unterschiedlichen Seiten der Ebene befinden, in der die Fläche $\mathcal{F}$ liegt.

Kante-Kante-Test

Der Kante-Kante-Test, den wir nun vorstellen, wurde im Rahmen des SiLVIA Projektes entwickelt und implementiert. Wenn wir analog vorgehen wollen, wie beim Ecke-Fläche-Test, haben wir das Problem, daß wir keine stetige vorzeichenbehaftete Distanz zwischen den Kanten definieren können. Zu einer Ecke und einer Fläche können wir klar anhand des Vorzeichens der Distanzfunktion entscheiden, ob die Ecke auf der „richtigen" oder auf der „falschen" Seite der Fläche liegt. Für zwei Kanten ist es schwierig, diese Entscheidung zu treffen. Aber wir machen folgende Beobachtung. Seien e_1 und e_2 zwei nicht parallele Kanten, und seien die beiden zu e_2 adjazenten Flächen nicht parallel. Wenn e_1 und e_2 echt miteinander kollidieren (echt heißt, daß der Kollisionspunkt nicht an einem Kantenende liegt), dann gibt es eine zu e_2 adjazente Fläche $\mathcal{F}$, so daß sich zum Kollisionszeitpunkt die beiden Endpunkte von e_1 auf verschiedenen Seiten der Ebene befinden, in der $\mathcal{F}$ liegt (Abbildung 4.4). Man beachte, daß dies nicht notwendig für beide zu e_2 adjazente Flächen gelten muß (Abbildung 4.5). Wir wollen diese Beobachtung als Lemma formulieren.

Lemma 4.1. *Sei eine Kante e durch ihre Endpunkte $\boldsymbol{a}, \boldsymbol{b}$ und eine Gerade g durch einen Punkt $\boldsymbol{p}$ und einen Richtungsvektor $\boldsymbol{u}$ gegeben. g und e seien nicht parallel, d.h. es gelte $\boldsymbol{u} \times (\boldsymbol{b} - \boldsymbol{a}) \neq \boldsymbol{0}$. Ferner seien zwei Ebenen $\mathcal{E}_1$ und $\mathcal{E}_2$ durch die Gleichungen $\boldsymbol{n}_1^T \boldsymbol{x} = \nu_1$ bzw. $\boldsymbol{n}_2^T \boldsymbol{x} = \nu_2$ gegeben und es gelte $\mathcal{E}_1 \cap \mathcal{E}_2 = g$. Die Kante e schneide die Gerade g echt, d.h. es gebe $\lambda, \mu \in \mathbb{R}, 0 < \lambda < 1$, so daß*

$$\boldsymbol{a} + \lambda(\boldsymbol{b} - \boldsymbol{a}) = \boldsymbol{p} + \mu \boldsymbol{u}$$

gilt. Dann gilt eine der folgenden beiden Aussagen:

$$\begin{array}{ll} \textit{(i)} & (\boldsymbol{n}_1^T \boldsymbol{a} - \nu_1) \cdot (\boldsymbol{n}_1^T \boldsymbol{b} - \nu_1) < 0 \\ \textit{(ii)} & (\boldsymbol{n}_2^T \boldsymbol{a} - \nu_2) \cdot (\boldsymbol{n}_2^T \boldsymbol{b} - \nu_2) < 0 \end{array}$$

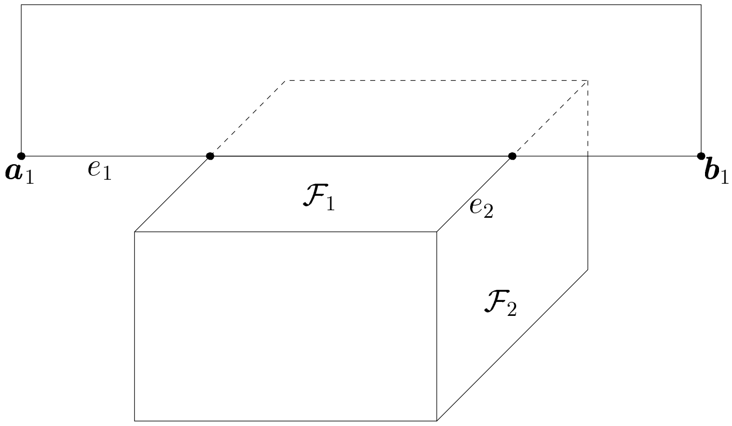

Abbildung 4.5: Die Kante e_1 kollidiert mit e_2. $\boldsymbol{a}_1$ und $\boldsymbol{b}_1$ liegen nicht auf verschiedenen Seiten von $\mathcal{F}_1$, da e_1 parallel zu dieser Fläche ist. Sie liegen aber auf verschiedenen Seiten von $\mathcal{F}_2$.

Beweis. Weil die Ebenen $\mathcal{E}_1$ und $\mathcal{E}_2$ sich in der Geraden g schneiden, muß gelten

$$\begin{aligned} \boldsymbol{n}_1^T \boldsymbol{p} = \nu_1, &\quad \boldsymbol{n}_1^T \boldsymbol{u} = 0 \\ \boldsymbol{n}_2^T \boldsymbol{p} = \nu_2, &\quad \boldsymbol{n}_2^T \boldsymbol{u} = 0. \end{aligned} \tag{4.2}$$

Wir beweisen die Behauptung des Lemmas indirekt und nehmen daher an, daß die Aussagen (i) und (ii) beide nicht gelten. Nach Voraussetzung gibt es $\lambda, \mu \in \mathbb{R}$ mit $0 < \lambda < 1$, so daß

$$\boldsymbol{a} + \lambda(\boldsymbol{b} - \boldsymbol{a}) = \boldsymbol{p} + \mu \boldsymbol{u}.$$

Durch Multiplikation von links mit $\boldsymbol{n}_1^T$ und Anwenden der ersten Zeile von (4.2) erhalten wir

$$(1 - \lambda)\boldsymbol{n}_1^T \boldsymbol{a} + \lambda \boldsymbol{n}_1^T \boldsymbol{b} = \nu_1.$$

Da laut Annahme die Aussage (i) nicht gilt, haben wir

$$\begin{array}{lcll} \boldsymbol{n}_1^T \boldsymbol{a} \geq \nu_1 & \wedge & \boldsymbol{n}_1^T \boldsymbol{b} \geq \nu_1 & \quad \text{oder} \\ \boldsymbol{n}_1^T \boldsymbol{a} \leq \nu_1 & \wedge & \boldsymbol{n}_1^T \boldsymbol{b} \leq \nu_1. & \end{array}$$

Ist eine dieser Ungleichungen echt, so folgt wegen $\lambda > 0$ und $1 - \lambda > 0$

$$\begin{array}{rcll} (1 - \lambda)\boldsymbol{n}_1^T \boldsymbol{a} + \lambda \boldsymbol{n}_1^T \boldsymbol{b} & > & \nu_1 & \quad \text{oder} \\ (1 - \lambda)\boldsymbol{n}_1^T \boldsymbol{a} + \lambda \boldsymbol{n}_1^T \boldsymbol{b} & < & \nu_1, & \end{array}$$

was ein Widerspruch ist. Also muß gelten:

$$\boldsymbol{n}_1^T \boldsymbol{a} = \boldsymbol{n}_1^T \boldsymbol{b} = \nu_1.$$

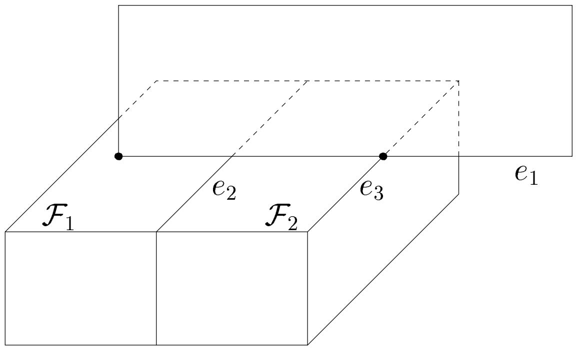

Abbildung 4.6: Beispiel für zwei zu e_2 adjazente parallele Flächen. Hier tritt eine Ecke-Fläche-Kollision und eine Kollision zwischen e_2 und e_3 auf, so daß eine Kollision zwischen e_1 und e_2 nicht erkannt werden muß.

Die Kante e liegt also in der Ebene $\mathcal{E}_1$. Auf analoge Weise benutzt man die Annahme, daß Aussage (ii) nicht gilt, um zu zeigen, daß e auch in $\mathcal{E}_2$ liegt, daß also gilt:

$$\boldsymbol{n}_2^T \boldsymbol{a} = \boldsymbol{n}_2^T \boldsymbol{b} = \nu_2.$$

Da die Schnittgerade zweier nicht paralleler Ebenen eindeutig ist, muß die Gerade durch $\boldsymbol{a}$ und $\boldsymbol{b}$ gleich g sein, d.h. es gilt $\boldsymbol{u} \times (\boldsymbol{b} - \boldsymbol{a}) = \boldsymbol{0}$. Dies ist ein Widerspruch zur Voraussetzung. $\square$

Wir wollen uns kurz überlegen, daß die Voraussetzungen von Lemma 4.1 vernünftig sind. Es wird zum einen vorausgesetzt, daß die beteiligten Kanten nicht parallel sind. Diese Annahme ist vernünftig, denn man überlegt sich leicht, daß bei der Kollision paralleler Kanten immer zwei Ecken beteiligt sind. Dieser Fall wird durch die Spezialfallbehandlung des Ecke-Fläche-Tests abgedeckt. Weiter wird vorausgesetzt, daß die beiden beteiligten Ebenen nicht parallel sind. Auch dies ist vernünftig. Denn wenn $\mathcal{E}_1$ und $\mathcal{E}_2$ parallel sind, gilt entweder $\boldsymbol{n}_1^T \boldsymbol{n}_2 = -|\boldsymbol{n}_1| \cdot |\boldsymbol{n}_2|$ oder $\boldsymbol{n}_1^T \boldsymbol{n}_2 = |\boldsymbol{n}_1| \cdot |\boldsymbol{n}_2|$. Der erste Fall wäre ein Widerspruch zu Punkt 3 unserer Definition eines Körpers in Abschnitt 3.2.1 und tritt somit nicht auf. Man überlegt sich, daß im zweiten Fall, wie in Abbildung 4.6 beispielhaft gezeigt, Ecke-Fläche-Kollisionen bzw. Kollisionen zwischen e_1 und Kanten $e \neq e_2$ stattfinden, so daß auch dieser Fall abgedeckt wird. Und schließlich ist auch die Annahme, daß die Kanten sich echt schneiden vernünftig, da ansonsten die Spezialfallbehandlung des Ecke-Fläche-Tests eine Kollision meldet.

Wir wenden nun das Lemma an, um einen Kollisionstest zu formulieren. Seien also eine Kante e_1 mit den Endpunkten $\boldsymbol{a}_1$ und $\boldsymbol{b}_1$ und eine Kante e_2 mit den Endpunkten $\boldsymbol{a}_2$ und $\boldsymbol{b}_2$ gegeben. Ferner sei zu e_2 eine Fläche adjazent, die in der Ebene mit der Gleichung $\boldsymbol{n}^T \boldsymbol{x} = \nu$ liegt. Wenn wir die Eckpunkte dieser Fläche im Gegenuhrzeigersinn um den Normalenvektor $\boldsymbol{n}$ aufzählen, soll $\boldsymbol{b}_2$ auf $\boldsymbol{a}_2$ folgen. Ansonsten werden $\boldsymbol{a}_2$ und $\boldsymbol{b}_2$ einfach vertauscht. Wir bestimmen nun alle maximalen Intervalle $I \subseteq [0, \Delta t]$, so daß für alle $t + \tau$ mit $\tau \in I$ die Punkte $\boldsymbol{a}_1$ und $\boldsymbol{b}_1$ auf verschiedenen Seiten der Ebene liegen. Falls

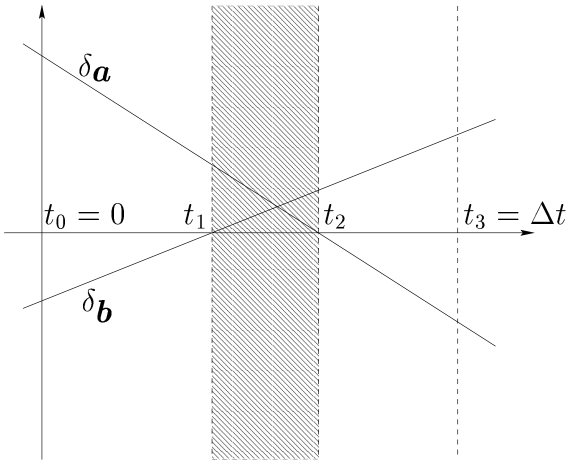

Abbildung 4.7: Beispiel für den Verlauf von $\delta_{\boldsymbol{a}}$ und $\delta_{\boldsymbol{b}}$. Der schraffierte Bereich ist ein Intervall, in dem beide Funktionen gleiches Vorzeichen haben. In diesem Intervall liegen die Punkte $\boldsymbol{a}_1$ und $\boldsymbol{b}_1$ nicht auf verschiedenen Seiten der Ebene.

ein solches Intervall nicht existiert, betrachten wir die zweite zu e_2 adjazente Fläche. Existiert auch für diese kein solches Intervall, garantiert Lemma 4.1, daß die Kanten nicht kollidieren. Wir bestimmen diese Intervalle durch lineare Interpolation. Zunächst definieren wir

$$\begin{aligned} \delta^t_{\boldsymbol{a}} &= (\boldsymbol{n}^t)^T \boldsymbol{a}^t_1 - \nu^t \\ \delta^t_{\boldsymbol{b}} &= (\boldsymbol{n}^t)^T \boldsymbol{b}^t_1 - \nu^t \end{aligned}$$

und analog $\delta^{t+\Delta t}_{\boldsymbol{a}}$ und $\delta^{t+\Delta t}_{\boldsymbol{b}}$. Dann definieren wir

$$\delta_{\boldsymbol{a}}(\tau) = \delta^t_{\boldsymbol{a}} + \frac{\tau}{\Delta t}(\delta^{t+\Delta t}_{\boldsymbol{a}} - \delta^t_{\boldsymbol{a}})$$

und analog $\delta_{\boldsymbol{b}}(\tau)$. Sei nun $k \leq 2$ die Anzahl der verschiedenen Nullstellen größer 0 und kleiner Δt von $\delta_{\boldsymbol{a}}$ und $\delta_{\boldsymbol{b}}$. Im Falle $k > 0$ seien t_i mit $1 \leq i \leq k$ diese Nullstellen und es gelte $t_i < t_j$ für $i < j$. Wir setzen $t_0 = 0$ und $t_{k+1} = \Delta t$. Die gesuchten maximalen Intervalle sind nun die Intervalle von t_i bis t_{i+1} mit $0 \leq i \leq k$, so daß

$$\delta_{\boldsymbol{a}}\left(\frac{t_i + t_{i+1}}{2}\right) \cdot \delta_{\boldsymbol{b}}\left(\frac{t_i + t_{i+1}}{2}\right) < 0.$$

Ein solches Intervall ist genau dann linksseitig abgeschlossen, wenn $\delta_{\boldsymbol{a}}(t_i) \cdot \delta_{\boldsymbol{b}}(t_i) \neq 0$ gilt (dann ist $i = 0$). Analog ist es genau dann rechtsseitig abgeschlossen, wenn $\delta_{\boldsymbol{a}}(t_{i+1}) \cdot \delta_{\boldsymbol{b}}(t_{i+1}) \neq 0$ gilt (dann ist $i = k$). In dem in Abbildung 4.7 gezeigten Beispiel ist $k = 2$. Die gesuchten Intervalle sind $[0, t_1)$ und $(t_2, \Delta t]$.

Wir führen nun für alle gefundenen Intervalle $I = (t_i, t_{i+1})$ einen Kollisionstest durch. (Wir schreiben der Einfachheit halber immer offene Intervalle, da dies im folgenden keine Rolle spielt.) Dabei beobachten wir, daß die Kanten e_1 und e_2 zu keiner Zeit $t + \tau$ mit $\tau \in I$ parallel sind, da e_1 die zu e_2 gehörende Ebene während des gesamten Intervalls immer in einem Punkt durchstößt. Dieser Durchstoßpunkt liegt, wie in [Cha95] gezeigt wird, genau dann auf der der adjazenten Fläche zugewandten Seite der Kante e_2, wenn gilt

$$\boldsymbol{n}^T(\boldsymbol{b}_1 - \boldsymbol{a}_1) \cdot \det[\boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_1 - \boldsymbol{a}_2] < 0. \tag{4.3}$$

Da e_1 und e_2 nicht parallel sind, ist

$$\frac{|\det[\boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_1 - \boldsymbol{a}_2]|}{|(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)|} \tag{4.4}$$

gerade die euklidische Distanz zwischen den Geraden durch e_1 und e_2. Es gibt nämlich $\alpha, \beta, \gamma \in \mathbb{R}$, so daß

$$\boldsymbol{a}_1 + \alpha(\boldsymbol{b}_1 - \boldsymbol{a}_1) + \gamma(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1) = \boldsymbol{a}_2 + \beta(\boldsymbol{b}_2 - \boldsymbol{a}_2) \tag{4.5}$$

ist. Dann ist $|\gamma| \cdot |(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)|$ die Distanz zwischen den Geraden. Man erhält durch Umformen dieser Gleichung und Anwenden der Cramerschen Regel

$$\begin{aligned} \gamma &= \frac{\det[\boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_2 - \boldsymbol{b}_2, \boldsymbol{a}_2 - \boldsymbol{a}_1]}{\det[\boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_2 - \boldsymbol{b}_2, (\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)]} \\ &= \frac{\det[\boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_2 - \boldsymbol{b}_2, \boldsymbol{a}_2 - \boldsymbol{a}_1]}{|(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)|^2}, \end{aligned}$$

woraus man (4.4) folgert. Wir können also (4.3) und (4.4) kombinieren, um für das gegebene Zeitintervall eine vorzeichenbehaftete Distanz zwischen e_1 und e_2 zu definieren. Wir setzen also

$$d^{t+t_i} = \operatorname{sign}(\boldsymbol{n}^T(\boldsymbol{b}_1 - \boldsymbol{a}_1)) \cdot \frac{\det[\boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_1 - \boldsymbol{a}_2]}{|(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)|}, \tag{4.6}$$

wobei wir für $\boldsymbol{a}_j, \boldsymbol{b}_j$ mit $j = 1, 2$ und für $\boldsymbol{n}$ die interpolierten Werte zum Zeitpunkt $t + t_i$ einsetzen. Analog definieren wir $d^{t+t_{i+1}}$ und setzen

$$d(\tau) = d^{t+t_i} + \frac{\tau - t_i}{t_{i+1} - t_i}(d^{t+t_{i+1}} - d^{t+t_i})$$

für $t_i \leq \tau \leq t_{i+1}$. Wie beim Ecke-Fläche-Test können wir nun ein Kollisionskriterium angeben:

- Falls $d(t_i) < -\varepsilon$ oder $d(t_i) \leq d(t_{i+1})$
  $\Longrightarrow$ keine Kollision.

- Ansonsten kollidieren die Geraden durch e_1 und e_2 zum Zeitpunkt

$$t_c = \begin{cases} t_i & \text{falls } d(t_i) \leq 0 \\ t_i + (t_{i+1} - t_i)\frac{d(t_i)}{d(t_i)-d(t_{i+1})} & \text{falls } d(t_i) > 0. \end{cases}$$

Wir entscheiden also, daß keine Kollision vorliegt, falls die Kante e_1 schon zum Zeitpunkt t_i auf der „falschen" Seite von e_2 liegt oder die Kanten sich voneinander entfernen. Ansonsten benutzen wir lineare Interpolation, um t_c so zu bestimmen, daß die Geraden durch e_1 und e_2 zum Zeitpunkt $t + t_c$ kollidieren. Wie beim Ecke-Fläche-Test verhindern wir, daß wir einen negativen Kollisionszeitpunkt erhalten, indem wir t_c auf Null setzen, wenn zum Zeitpunkt $t + t_i$ die Distanz negativ ist.

Nachdem wir festgestellt haben, daß die Geraden durch e_1 und e_2 kollidieren, müssen wir noch testen, ob der Kollisionspunkt auf beiden Kanten liegt. Dazu bestimmen wir in (4.5) α und β mit Hilfe der Cramerschen Regel, wobei wir für die $\boldsymbol{a}_j$ und $\boldsymbol{b}_j$ die interpolierten Werte zum Zeitpunkt $t + t_c$ einsetzen. Wir entscheiden dann, daß der Kollisionspunkt innerhalb beider Kanten liegt, wenn gilt

$$0 < \alpha, \beta < 1.$$

5 Berechnung von Kontaktkräften

In diesem Kapitel wollen wir Verfahren zur Berechnung von *Kontaktkräften* vorstellen, die für eine zwangsbasierte Bewegungssimulation benötigt werden. Unter Kontaktkräften verstehen wir die Kräfte, die starre Körper bei Kollisionen davon abhalten, sich gegenseitig zu durchdringen. Bei der Berechnung dieser Kräfte wollen wir in dieser Arbeit davon ausgehen, daß es keine Reibung zwischen den Objekten gibt.

Zunächst werden wir im ersten Abschnitt die Begriffe *Kontakt*, *Kontaktkraft* und *Kontaktdistanz* definieren. Im zweiten Abschnitt schildern wir dann ganz grob einen Algorithmus für zwangsbasierte Dynamiksimulation. Dieser Algorithmus wird eine Routine benutzen, die die Kontaktkräfte berechnet. Im dritten Abschnitt werden dann drei Verfahren zur Berechnung dieser Kräfte vorgestellt. Im Anschluß daran werden wir uns kurz die physikalischen Annahmen ansehen, die diesen drei Verfahren zugrunde liegen.

5.1 Kontakte, Kontaktkräfte und Kontaktdistanzen

Wir werden für die Berechnung von Kontaktkräften die Begriffe *Kontakt*, *Kontaktkraft* und *Kontaktdistanz* brauchen, die wir jetzt einführen wollen.

Ein *Kontakt* zwischen den Objekten $\mathcal{O}_i$ und $\mathcal{O}_j$ ist formal ein Sechstupel

$$(i, j, \mathcal{X}_1, \mathcal{X}_2, \boldsymbol{p}, \boldsymbol{n}).$$

Dabei heißt $\boldsymbol{p}$ der *Kontaktpunkt* und $\boldsymbol{n}$ die *Kontaktnormale* und es gilt eine der folgenden drei Bedingungen:

1. $\mathcal{X}_1$ ist eine Ecke von $\mathcal{O}_i$ und $\mathcal{X}_2$ eine Fläche von $\mathcal{O}_j$, und $\boldsymbol{n}$ ist die Normale von $\mathcal{X}_2$.
2. $\mathcal{X}_1$ ist eine Fläche von $\mathcal{O}_i$ und $\mathcal{X}_2$ eine Ecke von $\mathcal{O}_j$, und $-\boldsymbol{n}$ ist die Normale von $\mathcal{X}_1$.
3. $\mathcal{X}_1$ ist eine Kante von $\mathcal{O}_i$ und $\mathcal{X}_2$ eine Kante von $\mathcal{O}_j$. Sind $\boldsymbol{a}_1, \boldsymbol{b}_1$ die Endpunkte von $\mathcal{X}_1$ und $\boldsymbol{a}_2, \boldsymbol{b}_2$ die Endpunkte von $\mathcal{X}_2$, und sind $\boldsymbol{n}_1$ und $\boldsymbol{n}_2$ die Normalen der beiden zu $\mathcal{X}_2$ adjazenten Flächen, dann ist $\boldsymbol{n} = \sigma \frac{\boldsymbol{n}'}{|\boldsymbol{n}'|}$. Dabei ist $\boldsymbol{n}' = (\boldsymbol{b}_1 - \boldsymbol{a}_1) \times (\boldsymbol{b}_2 - \boldsymbol{a}_2)$ und $\sigma = \operatorname{sign}(\boldsymbol{n}'^T(\boldsymbol{n}_1 + \boldsymbol{n}_2))$.

In den ersten beiden Fällen nennen wir den Kontakt einen *Ecke-Fläche-Kontakt*, im dritten Fall einen *Kante-Kante-Kontakt*. Die intuitive Bedeutung eines Kontaktes $\kappa = (i, j, \mathcal{X}_1, \mathcal{X}_2, \boldsymbol{p}, \boldsymbol{n})$ ist, daß sich die Objekte $\mathcal{O}_i$ und $\mathcal{O}_j$ mit den Objektteilen $\mathcal{X}_1$ und $\mathcal{X}_2$ im Punkt $\boldsymbol{p}$ so berühren, daß Kontaktkräfte in Richtung der Vektoren $\boldsymbol{n}$ und $-\boldsymbol{n}$ wirken müssen, um eine Durchdringung der Objekte zu verhindern. Bei einem Kontakt zwischen einer Ecke und einer Fläche wirken diese Kräfte im reibungsfreien Fall parallel zur Flächennormalen und bei einem Kontakt zwischen Kanten parallel zum Kreuzprodukt der Kantenrichtungen. Der Grund für die Multiplikation mit σ bei der Berechnung der Kontaktnormalen im Kante-Kante-Fall ist, daß wir immer gewährleisten wollen, daß der Vektor $\boldsymbol{n}$ lokal im Punkt $\boldsymbol{p}$ von Objekt $\mathcal{O}_j$ zu Objekt $\mathcal{O}_i$ zeigt. Abbildung 5.1 zeigt eine

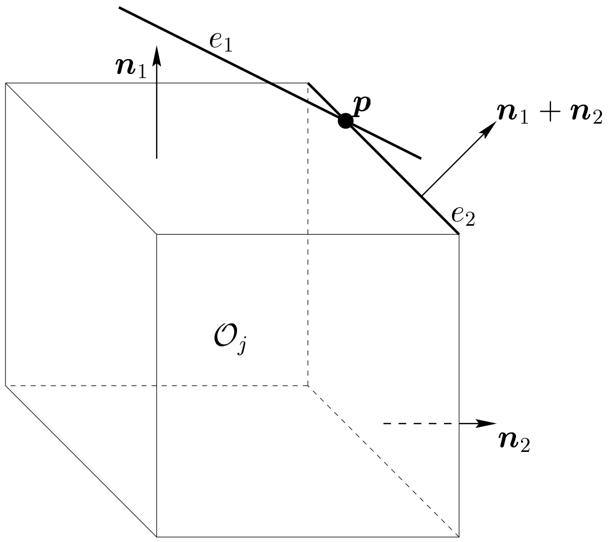

Abbildung 5.1: Beispielsituation bei einem Kante-Kante-Kontakt

beispielhafte Situation für einen Kante-Kante-Kontakt $(i, j, e_1, e_2, \boldsymbol{p}, \boldsymbol{n})$. Da $\boldsymbol{n}_1$ und $\boldsymbol{n}_2$ als Normalenvektoren normiert sind und somit gleiche Länge haben, zeigt der Vektor $\boldsymbol{n}_1 + \boldsymbol{n}_2$ in Richtung der Winkelhalbierenden zwischen $\boldsymbol{n}_1$ und $\boldsymbol{n}_2$ und damit von Objekt $\mathcal{O}_j$ weg. Man überlegt sich, daß σ genau dann negativ wird, wenn $\boldsymbol{n}'$ in die falsche Richtung zeigt, d.h. von $\mathcal{O}_i$ zu $\mathcal{O}_j$.

Die *Kontaktkraft* im Kontakt κ ist, wie schon erwähnt, die Kraft, die im Kontaktpunkt $\boldsymbol{p}$ die Objekte $\mathcal{O}_i$ und $\mathcal{O}_j$ in Richtung der Kontaktnormalen $\boldsymbol{n}$ auseinanderdrücken muß, um eine Durchdringung zu verhindern. Ist also f die Größe dieser Kraft, so wirkt die Kraft $f\boldsymbol{n}$ auf $\mathcal{O}_i$ und die Kraft $-f\boldsymbol{n}$ auf $\mathcal{O}_j$, da die Kontaktnormale im Kontaktpunkt von $\mathcal{O}_j$ nach $\mathcal{O}_i$ zeigt.

Während einer Bewegungssimulation liegt zu jedem Zeitpunkt eine Menge $\mathcal{K}$ von Kontakten vor. In Abbildung 5.2 ist eine Situation mit der Kontaktmenge

$$\begin{array}{ll} \{ & (1, 2, \ldots, \ldots, \boldsymbol{p}_1, \boldsymbol{n}_1), \\ & (3, 2, \ldots, \ldots, \boldsymbol{p}_2, \boldsymbol{n}_2), \\ & (1, 3, \ldots, \ldots, \boldsymbol{p}_3, \boldsymbol{n}_3), \\ & (4, 3, \ldots, \ldots, \boldsymbol{p}_4, \boldsymbol{n}_4) \quad \} \end{array}$$

dargestellt. Da die Darstellung zweidimensional ist, haben wir die Objektteile nicht benannt, sondern an den entsprechenden Stellen Punkte gesetzt. Es ist nicht immer so,

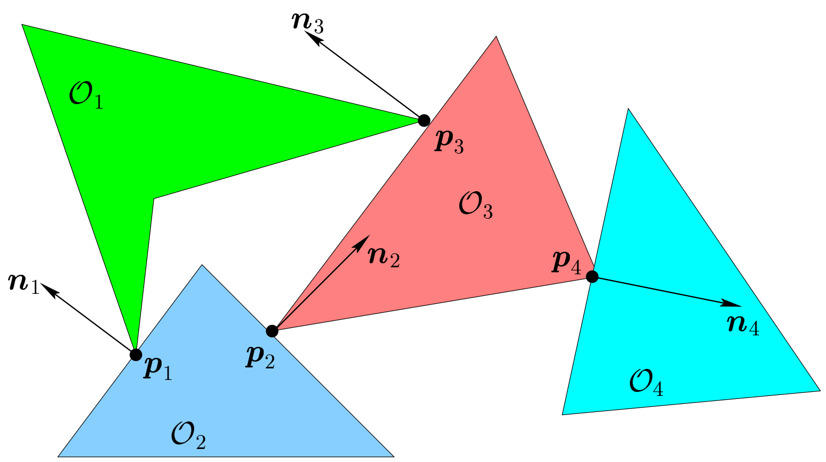

Abbildung 5.2: Beispiel für eine Menge von vier Kontakten

daß die Objekte sich im Kontaktpunkt tatsächlich berühren. Dies liegt daran, daß wir mit diskreten Zeitpunkten rechnen. Zum Zeitpunkt 0, also zu Beginn der Simulation, ist $\mathcal{K} = \emptyset$. Wir beschreiben nun, wie sich diese Menge bei der Simulation des Zeitintervalls $[t, t + \Delta t]$ ändert. Dabei wollen wir folgende Invariante garantieren:

> Zu Beginn der Simulation eines Zeitintervalls gilt für jeden Kontakt $(i, j, \mathcal{X}_1, \mathcal{X}_2, \boldsymbol{p}, \boldsymbol{n}) \in \mathcal{K}$, daß sich die Objektteile $\mathcal{X}_1$ und $\mathcal{X}_2$ berühren.

Sei $K = |\mathcal{K}|$, und sei $\mathcal{K} = \{\kappa_k | 1 \leq k \leq K\}$ mit $\kappa_k = (i_k, j_k, \mathcal{X}_{1k}, \mathcal{X}_{2k}, \boldsymbol{p}_k, \boldsymbol{n}_k)$. Wir nennen einen Kontakt κ_k einen *alten* Kontakt, wenn schon am Ende der Simulation des letzten Zeitintervalls ein Kontakt $(i_k, j_k, \mathcal{X}_{1k}, \mathcal{X}_{2k}, \boldsymbol{p}', \boldsymbol{n}')$ vorlag. Alle anderen Kontakte sind erst während der Simulation dieses Zeitintervalls neu definiert worden, und werden *neue* Kontakte genannt. Wann immer während der Simulation des Zeitintervalls $[t, t+\Delta t]$ die Kollisionserkennung eine Kollision zwischen den Objekten $\mathcal{O}_i$ und $\mathcal{O}_j$ zum Zeitpunkt t_{col} mit den Objektteilen $\mathcal{X}_1$ und $\mathcal{X}_2$ und dem Kollisionspunkt $\boldsymbol{p}$ meldet, definieren wir den Kontakt

$$\kappa = (i, j, \mathcal{X}_1, \mathcal{X}_2, \boldsymbol{p}, \boldsymbol{n}).$$

Dabei benutzen wir bei der Berechnung von $\boldsymbol{n}$ die (interpolierten) Objektdaten zum Zeitpunkt t_{col}. Für $k = 1, \ldots, K$ setzen wir $\kappa'_k = \kappa_k$, falls κ_k ein neuer Kontakt ist. Ist κ_k ein alter Kontakt, so setzen wir $\kappa'_k = (i_k, j_k, \mathcal{X}_{1k}, \mathcal{X}_{2k}, \boldsymbol{p}'_k, \boldsymbol{n}'_k)$. Dabei benutzen wir für die Berechnung von $\boldsymbol{n}'_k$ die Objektdaten zum Zeitpunkt t, und $\boldsymbol{p}'_k$ ist der Berührpunkt von $\mathcal{X}_{1k}$ und $\mathcal{X}_{2k}$. Dieser Berührpunkt existiert, da die Invariante gilt. Wir setzen nun

$$\mathcal{K}' = \{\kappa\} \cup \{\kappa'_k | 1 \leq k \leq K\}$$

und fahren mit der Simulation fort. Am Ende der Simulation des Intervalls stehen die neuen Objektkonfigurationen fest und die Objekte werden in diese Konfigurationen gebracht. Danach werden diejenigen Ecke-Fläche-Kontakte aus der Kontaktmenge entfernt, für die sich die Ecke und die Fläche nicht mehr berühren. Ferner werden alle Kante-Kante-Kontakte entfernt, für die die Kanten sich nicht mehr echt schneiden. Es ist nun klar, daß nach dem Entfernen von Kontakten, in denen sich die Objekte nicht berühren, die Invariante wieder gilt. Also gilt sie zu Beginn des nächsten Simulationsschrittes.

Nun wollen wir noch den Begriff *Kontaktdistanz* einführen. Es seien eine Konfiguration $\boldsymbol{s}^t$ der Welt zum Zeitpunkt t und eine gewünschte Konfiguration $\boldsymbol{s}^{t+\Delta t}$ zum Zeitpunkt $t+\Delta t$ gegeben. Außerdem bestehe ein Kontakt $\kappa = (i, j, \mathcal{X}_1, \mathcal{X}_2, \boldsymbol{p}, \boldsymbol{n})$. Unter der Kontaktdistanz w^t bzw. $w^{t+\Delta t}$ verstehen wir die vorzeichenbehaftete euklidische Distanz zwischen den Objektteilen $\mathcal{X}_1$ und $\mathcal{X}_2$ bei Konfiguration $\boldsymbol{s}^t$ bzw. $\boldsymbol{s}^{t+\Delta t}$. Wie wir bereits gesehen haben, ist während der Simulation des Intervalls $[t, t+\Delta t]$ die Kontaktdistanz w^t für alle alten Kontakte gleich Null, da die Invariante gilt. Für die neuen Kontakte ist diese Distanz größer oder gleich Null, da die Konfiguration zum Zeitpunkt t immer legal ist. Das Ziel der Kontaktkraftberechnung wird es sein, die Kontaktkräfte so zu bestimmen, daß eine legale Konfiguration $\boldsymbol{s}^{t+\Delta t}$ entsteht, d.h. daß die Kontaktdistanzen $w^{t+\Delta t}$ für alle Kontakte größer gleich Null sind. Daher wollen wir zeigen, wie man die Kontaktdistanzen zum Zeitpunkt $t+\Delta t$ in Abhängigkeit von den wirkenden Kräften berechnet. Dazu überlegen wir uns zunächst, wie man diese Distanzen in Abhängigkeit von den Objektdaten zum Zeitpunkt $t+\Delta t$ erhält.

Ecke-Fläche-Kontakt:

Sei ein Kontakt $(i, j, \boldsymbol{a}, \mathcal{F}, \boldsymbol{p}_\kappa, \boldsymbol{n}_\kappa)$ zwischen einer Ecke $\boldsymbol{a}$ von Objekt $\mathcal{O}_i$ und einer Fläche $\mathcal{F}$ von Objekt $\mathcal{O}_j$ gegeben. $\mathcal{F}$ liege in der Ebene mit der Gleichung $\boldsymbol{n}^T\boldsymbol{x} = n_0$. Dann gilt

$$w^{t+\Delta t} = (\boldsymbol{n}^{t+\Delta t})^T \boldsymbol{a}^{t+\Delta t} - n_0^{t+\Delta t}. \tag{5.1}$$

Kante-Kante-Kontakt:

Sei nun ein Kontakt $(i, j, e_1, e_2, \boldsymbol{p}_\kappa, \boldsymbol{n}_\kappa)$ zwischen einer Kante e_1 von Objekt $\mathcal{O}_i$ und einer Kante e_2 von Objekt $\mathcal{O}_j$ gegeben. Die Endpunkte der Kanten seien $\boldsymbol{a}_1, \boldsymbol{b}_1$ bzw. $\boldsymbol{a}_2, \boldsymbol{b}_2$. Wir würden nun gerne wie bei der Kollisionserkennung die Formel (4.6) benutzen, um die vorzeichenbehaftete Distanz der beiden Kanten zum Zeitpunkt $t+\Delta t$ zu berechnen. Dies können wir aber nicht in allen Fällen tun, da es zu diesem Zeitpunkt evtl. keine zu e_2 adjazente Fläche gibt mit $(\boldsymbol{n}^{t+\Delta t})^T(\boldsymbol{b}_1^{t+\Delta t} - \boldsymbol{a}_1^{t+\Delta t}) \neq 0$, d.h. die Kanten könnten zu diesem Zeitpunkt parallel sein. Um die euklidische Distanz paralleler Geraden zu berechnen, betrachten wir Abbildung 5.3. Die Distanz zwischen e_1 und e_2 ist gleich der Distanz zwischen dem Punkt $\boldsymbol{a}_1$ und der Ebene $\mathcal{E}$. Diese Ebene genügt der Gleichung $\boldsymbol{n}^T\boldsymbol{x} = \boldsymbol{n}^T\boldsymbol{a}_2$, wobei wir $\boldsymbol{n} = [(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{a}_1 - \boldsymbol{a}_2)] \times (\boldsymbol{b}_2 - \boldsymbol{a}_2)$ gesetzt haben. Die Distanz zwischen $\boldsymbol{a}_1$ und $\mathcal{E}$ ist dann

$$\begin{aligned} d_{\|}^{t+\Delta t} &= \frac{\boldsymbol{n}^T(\boldsymbol{a}_1 - \boldsymbol{a}_2)}{|\boldsymbol{n}|} \\ &= \frac{\det[(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{a}_1 - \boldsymbol{a}_2), \boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{a}_1 - \boldsymbol{a}_2]}{|[(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{a}_1 - \boldsymbol{a}_2)] \times (\boldsymbol{b}_2 - \boldsymbol{a}_2)|}. \end{aligned}$$

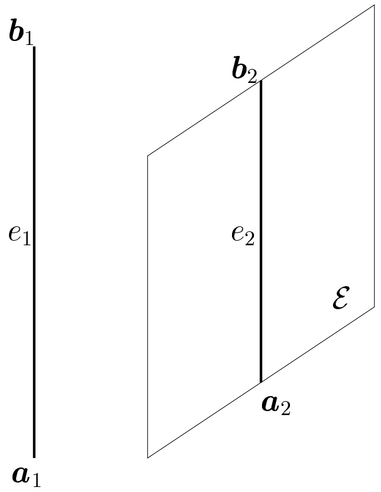

Abbildung 5.3: Veranschaulichung zur Berechnung des Abstandes paralleler Geraden.

Das Vorzeichen dieser Distanz bestimmen wir mit Hilfe der Kontaktnormalen $\boldsymbol{n}_\kappa$, die nach Definition von $\mathcal{O}_j$ nach $\mathcal{O}_i$ zeigt. Wir setzen also

$$\sigma_{||}^{t+\Delta t} = \boldsymbol{n}_\kappa^T(\boldsymbol{a}_1 - \boldsymbol{a}_2).$$

Bei der Berechnung von $d_{||}^{t+\Delta t}$ und $\sigma_{||}^{t+\Delta t}$ setzen wir $\boldsymbol{a}_1, \boldsymbol{b}_1, \boldsymbol{a}_2$ und $\boldsymbol{b}_2$ zum Zeitpunkt $t + \Delta t$ ein. Wir definieren nun

$$w_{||}^{t+\Delta t} = \sigma_{||}^{t+\Delta t} \cdot d_{||}^{t+\Delta t}.$$

Für den Fall, daß e_1 und e_2 zum Zeitpunkt $t + \Delta t$ nicht parallel sind, benutzen wir (4.6). Sei dazu $\boldsymbol{n}$ die Normale einer zu e_2 adjazenten Fläche, für die $(\boldsymbol{n}^{t+\Delta t})^T(\boldsymbol{b}_1^{t+\Delta t} - \boldsymbol{a}_1^{t+\Delta t}) \neq 0$ gilt. Dann definieren wir

$$w_{\nparallel}^{t+\Delta t} = \operatorname{sign}(\boldsymbol{n}^T(\boldsymbol{b}_1 - \boldsymbol{a}_1)) \cdot \frac{\det[\boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{a}_1 - \boldsymbol{a}_2]}{|(\boldsymbol{b}_2 - \boldsymbol{a}_2) \times (\boldsymbol{b}_1 - \boldsymbol{a}_1)|}.$$

Auch hier setzen wir die Werte zum Zeitpunkt $t + \Delta t$ ein. Unsere Definition für die Kontaktdistanz $w^{t+\Delta t}$ ist also

$$w^{t+\Delta t} = \begin{cases} w_{\nparallel}^{t+\Delta t} & \text{falls } (\boldsymbol{b}_2^{t+\Delta t} - \boldsymbol{a}_2^{t+\Delta t}) \times (\boldsymbol{b}_1^{t+\Delta t} - \boldsymbol{a}_1^{t+\Delta t}) \neq \boldsymbol{0}, \\ w_{||}^{t+\Delta t} & \text{sonst.} \end{cases} \tag{5.2}$$

Mit (5.1) und (5.2) können wir für jeden Kontakt eine Kontaktdistanz $w^{t+\Delta t}$ berechnen. Die Formeln enthalten Punktkoordinaten $\boldsymbol{a}^{t+\Delta t}$ und Koeffizienten $\boldsymbol{n}^{t+\Delta t}, n_0^{t+\Delta t}$ von

Ebenengleichungen. Für diese Werte können wir (3.23), (3.24) und (3.25) einsetzen:

$$\begin{aligned}\boldsymbol{a}^{t+\Delta t} &= \mathbf{R}^{t+\Delta t}\hat{\boldsymbol{a}} + \boldsymbol{c}^{t+\Delta t}\\ \boldsymbol{n}^{t+\Delta t} &= \mathbf{R}^{t+\Delta t}\hat{\boldsymbol{n}}\\ n_0^{t+\Delta t} &= \hat{n}_0 + (\boldsymbol{c}^{t+\Delta t})^T\mathbf{R}^{t+\Delta t}\hat{\boldsymbol{n}}.\end{aligned}$$

Somit erhalten wir Formeln für die Kontaktdistanzen, die von den Objektkonfigurationen $\mathbf{R}^{t+\Delta t}$ und $\boldsymbol{c}^{t+\Delta t}$ abhängen. Für diese können wir die diskretisierten Bewegungsgleichungen (3.34), (3.38) und (3.35) einsetzen:

$$\begin{aligned}\boldsymbol{c}^{t+\Delta t} &= \boldsymbol{c}^t + \Delta t\boldsymbol{v}^{t+\Delta t}\\ \mathbf{R}^{t+\Delta t} &= \mathbf{R}(\mathfrak{q}^{t+\Delta t})\\ \mathfrak{q}^{t+\Delta t} &= \mathfrak{q}^t + \frac{1}{2}\Delta t\boldsymbol{\omega}^{t+\Delta t}\cdot\mathfrak{q}^t.\end{aligned}$$

Um $\mathbf{R}(\mathfrak{q}^{t+\Delta t})$ zu bestimmen, normieren wir das Quaternion $\mathfrak{q}^{t+\Delta t}$ und benutzen die Formel (3.9):

$$\mathbf{R}(\mathfrak{q}) = (q_0^2 - \boldsymbol{q}^T\boldsymbol{q})\mathbf{E} + 2\boldsymbol{q}\boldsymbol{q}^T + 2q_0\boldsymbol{q}^\times.$$

Nun haben wir also Formeln für die Kontaktdistanzen, die von den momentanen Objektkonfigurationen $\mathfrak{q}^t$ und $\boldsymbol{c}^t$ und von den Geschwindigkeiten $\boldsymbol{v}^{t+\Delta t}$ und $\boldsymbol{\omega}^{t+\Delta t}$ abhängen. Die momentanen Objektkonfigurationen kennen wir aus dem letzten Simulationsschritt. Die Geschwindigkeiten ergeben sich durch die Kontaktkräfte und die diskretisierten Bewegungsgleichungen (3.36) und (3.37). Um diese Gleichungen anwenden zu können, müssen wir für jedes Objekt $\mathcal{O}_i$ wissen, welche Kräfte auf dieses Objekt wirken. Seien also die Kontakte $\kappa_1, \ldots, \kappa_K$ gegeben mit $\kappa_k = (i_k, j_k, \mathcal{X}_{1k}, \mathcal{X}_{2k}, \boldsymbol{p}_k, \boldsymbol{n}_k)$ für $1 \leq k \leq K$. Im Kontakt κ_k wirke eine Kontaktkraft der Größe f_k. Wir definieren die Vektoren $\boldsymbol{r}_{kl} = \boldsymbol{p}_k - \boldsymbol{c}_l^t$ vom Schwerpunkt von Objekt $\mathcal{O}_l$ zum Kontaktpunkt $\boldsymbol{p}_k$. Wie wir bereits gesehen haben, wirkt im Kontakt κ_k die Kraft $f_k\boldsymbol{n}_k$ auf Objekt $\mathcal{O}_{i_k}$ und die Kraft $-f_k\boldsymbol{n}_k$ auf Objekt $\mathcal{O}_{j_k}$. Auf ein gegebenes Objekt $\mathcal{O}_i$ wirken also die Kräfte $f_k\boldsymbol{n}_k$ mit $i_k = i$ und die Kräfte $-f_k\boldsymbol{n}_k$ mit $j_k = i$. Damit erhalten wir aus den Bewegungsgleichungen (3.36) und (3.37) für die Geschwindigkeiten $\boldsymbol{v}_i^{t+\Delta t}$ und $\boldsymbol{\omega}_i^{t+\Delta t}$ von Objekt $\mathcal{O}_i$:

$$\boldsymbol{v}_i^{t+\Delta t} = \boldsymbol{v}_i^t + \frac{\Delta t}{m_i}\left(\sum_{k\in\{k|i_k=i\}} f_k\boldsymbol{n}_k - \sum_{k\in\{k|j_k=i\}} f_k\boldsymbol{n}_k\right) \tag{5.3}$$

$$\begin{aligned}\boldsymbol{\omega}_i^{t+\Delta t} = \boldsymbol{\omega}_i^t + \Delta t(\mathbf{I}_i^t)^{-1}\Bigg(&\sum_{k\in\{k|i_k=i\}} f_k(\boldsymbol{r}_{ki}\times\boldsymbol{n}_k)\\ &- \sum_{k\in\{k|j_k=i\}} f_k(\boldsymbol{r}_{ki}\times\boldsymbol{n}_k) - \boldsymbol{\omega}_i^t\times\mathbf{I}_i^t\boldsymbol{\omega}_i^t\Bigg).\end{aligned} \tag{5.4}$$

Setzen wir diese Gleichungen in unsere Formeln für die Kontaktdistanzen ein, so hängen die $w^{t+\Delta t}$ nur noch von den momentanen Objektkonfigurationen und den Kontaktkräften ab.

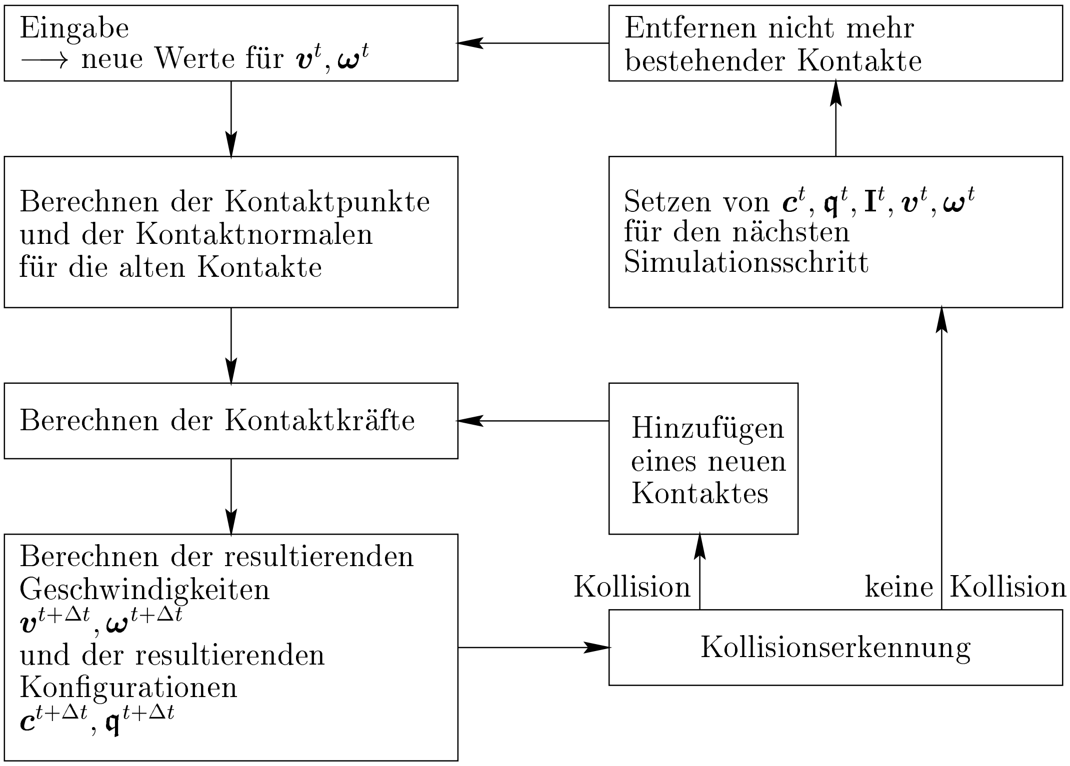

Abbildung 5.4: Arbeitsweise eines Algorithmus zur Bewegungssimulation mittels Kontaktkräften

5.2 Ein grober Algorithmus zur Bewegungssimulation

In diesem Abschnitt wollen wir an einem groben Algorithmus erläutern, wie eine Bewegungssimulation mittels Kontaktkräften arbeitet. Wir tun dies anhand von Abbildung 5.4, die die Arbeitsweise des Algorithmus schematisch beschreibt. Dieser verwendet Konfigurationsvariablen $\boldsymbol{c}^t, \boldsymbol{c}^{t+\Delta t}, \mathbf{q}^t, \mathbf{q}^{t+\Delta t}$ für jedes Objekt. Ferner verwaltet er für jedes Objekt die Trägheitsmatrix $\mathbf{I}^t$ und Variablen $\boldsymbol{v}^t, \boldsymbol{v}^{t+\Delta t}, \boldsymbol{\omega}^t, \boldsymbol{\omega}^{t+\Delta t}$ für die Geschwindigkeiten. Zu Beginn seien folgende Variablen bereits initialisiert:

- die aktuellen Objektkonfigurationen: $\boldsymbol{c}^t, \mathbf{q}^t$
- die aktuellen Trägheitsmatrizen der Objekte: $\mathbf{I}^t$
- die aktuellen Objektgeschwindigkeiten: $\boldsymbol{v}^t, \boldsymbol{\omega}^t$.

Außerdem wird eine Menge von Kontakten verwaltet, die zu Beginn leer ist. Nachdem der Algorithmus gestartet wurde, werden durch eine Eingabe neue Objektgeschwindigkeiten $\boldsymbol{v}^t, \boldsymbol{\omega}^t$ bestimmt. Unter „Eingabe“ verstehen wir dabei sowohl die Eingabe des Benutzers (z.B. per Maus), die wir entweder direkt als Geschwindigkeiten oder als Kräfte interpretieren können, als auch von einem übergeordneten Algorithmus gemachte Bewegungsvorgaben und von „außen“ wirkende Kräfte, wie z.B. die Gravitationskraft. Anschließend werden für alle *alten* Kontakte die Kontaktpunkte $\boldsymbol{p}$ und die Kontaktnormalen $\boldsymbol{n}$ berechnet, wie im letzten Abschnitt beschrieben. (Beim ersten Durchlauf passiert hier natürlich noch nichts, da die Kontaktmenge dann leer ist.) Danach werden die Kontaktkräfte für alle Kontakte berechnet. Wir werden im nächsten Abschnitt drei Verfahren vorstellen, mit deren Hilfe man diese Berechnung durchführen kann. Nachdem die Kontaktkräfte bekannt sind, werden die resultierenden Geschwindigkeiten $\boldsymbol{v}^{t+\Delta t}$ und $\boldsymbol{\omega}^{t+\Delta t}$ für alle Objekte gemäß den Gleichungen (5.3) und (5.4) berechnet und schließlich die resultierenden Objektkonfigurationen $\boldsymbol{c}^{t+\Delta t}$ und $\mathbf{q}^{t+\Delta t}$ mittels der Bewegungsgleichungen (3.34) und (3.35) ermittelt. Nun starten wir den Kollisionserkennungsalgorithmus, um festzustellen, ob diese resultierenden Konfigurationen eine legale Konfiguration der Welt sind, oder ob eine Kollision stattfindet. Dabei nehmen wir diejenigen Paare von Objektteilen, zwischen denen bereits ein Kontakt besteht, von der Kollisionserkennung aus. Falls keine Kollision stattfindet, sind wir mit der Simulation des Bewegungsschrittes fast fertig. Es werden nur noch die Konfigurationsvariablen $\boldsymbol{c}^t, \mathbf{q}^t$, die Trägheitsmatrizen $\mathbf{I}^t$ und die Geschwindigkeitsvariablen $\boldsymbol{v}^t, \boldsymbol{\omega}^t$ aktualisiert und diejenigen Kontakte aus der Kontaktmenge entfernt, in denen die entsprechenden Objektteile sich nun nicht mehr berühren. Danach beginnt der Algorithmus von vorne, um den nächsten Bewegungsschritt zu simulieren. Meldet das Kollisionserkennungsverfahren dagegen eine Kollision, so wird, wie im letzten Abschnitt beschrieben, die Kontaktmenge um einen neuen Kontakt erweitert. Danach fährt der Algorithmus wieder mit der Berechnung der Kontaktkräfte fort. Dies wird solange wiederholt, bis keine neue Kollision mehr stattfindet. Dies ist früher oder später der Fall, da bei der Kollisionserkennung die bereits in Kontakt stehenden Objektteilpaare ausgenommen werden und es in einer Welt nur endlich viele Objektteilpaare gibt, die miteinander in Kontakt sein können.

Dieser Algorithmus benötigt also ein Verfahren zur Berechnung der Kontaktkräfte. Wir wollen im nun folgenden Abschnitt drei solche Verfahren ausführlich beschreiben.

5.3 Drei Verfahren zur Berechnung von Kontaktkräften

Für die Verfahren, die wir im folgenden vorstellen werden, sei folgende Situation gegeben: Unsere Welt sei die Menge $\{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$. Die momentanen Objektkonfigurationen seien $\boldsymbol{c}_1^t, \mathbf{q}_1^t, \ldots, \boldsymbol{c}_N^t, \mathbf{q}_N^t$. Ferner liegen die Kontakte $\kappa_1, \ldots, \kappa_K$ vor, wobei $\kappa_k = (i_k, j_k, \mathcal{X}_{1k}, \mathcal{X}_{2k}, \boldsymbol{p}_k, \boldsymbol{n}_k)$ sei für $1 \le k \le K$. Außerdem definieren wir wie bei der Herleitung der Gleichungen (5.3) und (5.4) für $1 \le k \le K$ und $1 \le l \le N$ die Vektoren $\boldsymbol{r}_{kl} = \boldsymbol{p}_k - \boldsymbol{c}_l^t$, die jeweils vom Schwerpunkt von $\mathcal{O}_l$ zum Kontaktpunkt $\boldsymbol{p}_k$ zeigen. Ferner seien $w_k : \mathbb{R}^K \to \mathbb{R}$ die Funktionen, die gemäß Abschnitt 5.1 für Kontaktkräfte $f_1, \ldots, f_K \in \mathbb{R}$ die daraus resultierende Kontaktdistanz im k-ten Kontakt zum Zeitpunkt $t + \Delta t$ berechnet. Um übersichtlichere Notationen zu haben, definieren wir die Funktion

$\boldsymbol{w} : \mathbb{R}^K \to \mathbb{R}^K$ mit

$$\boldsymbol{w}(\boldsymbol{f}) = \begin{pmatrix} w_1(f_1, \ldots, f_K) \\ w_2(f_1, \ldots, f_K) \\ \vdots \\ w_K(f_1, \ldots, f_K) \end{pmatrix}$$

für $\boldsymbol{f} = (f_1, \ldots, f_K)^T \in \mathbb{R}^K$.

5.3.1 Formulierung als nichtlineares Gleichungssystem

Das erste Verfahren, das wir vorstellen wollen, formuliert das Problem, die Kontaktkräfte zu bestimmen, als nichtlineares Gleichungssystem. Es wird einfach gefordert, daß zum Zeitpunkt $t + \Delta t$ in allen bestehenden Kontakten die Kontaktdistanz gleich Null wird. Mit obiger Notation lassen sich diese Zwangsbegingungen schreiben als

$$\boldsymbol{w}(\boldsymbol{f}) = \boldsymbol{0}. \tag{5.5}$$

Dies ist ein nichtlineares Gleichungssystem mit den Kontaktkräften $f_1, \ldots, f_K$ als Unbekannten. Zum Lösen dieses Gleichungssystems können wir z.B. das folgende Verfahren (Newton-Raphson-Verfahren) verwenden.
Wir beginnen mit einem Startvektor $\boldsymbol{f}^{(0)}$. Für $j \geq 0$ bestimmen wir $\Delta\boldsymbol{f}^{(j)}$ als Lösung des linearen Gleichungssystems

$$\frac{d\boldsymbol{w}(\boldsymbol{f}^{(j)})}{d\boldsymbol{f}} \cdot \Delta\boldsymbol{f}^{(j)} = -\boldsymbol{w}(\boldsymbol{f}^{(j)})$$

und setzen $\boldsymbol{f}^{(j+1)} = \boldsymbol{f}^{(j)} + \Delta\boldsymbol{f}^{(j)}$. Dies iterieren wir solange, bis $|\boldsymbol{w}(\boldsymbol{f}^{(j)})|$ unter eine vorgegebene Schranke fällt. Der zuletzt berechnete Lösungsvektor $\boldsymbol{f}^{(j)}$ ist dann unsere gesuchte Lösung. Als Startvektor $\boldsymbol{f}^{(0)}$ bietet sich der Lösungsvektor $\boldsymbol{f}$ des vorangegangenen Simulationsschrittes an. Aufgrund numerischer Ungenauigkeiten (Linearisierungen bei der Kollisionserkenung, Diskretisierung der Bewegungsgleichungen) kann es passieren, daß wiedersprüchliche Kontakte auftreten, so daß das Gleichungssystem (5.5) gar keine Lösung hat. Ein anderes Problem kann auftreten, wenn man redundante Kontakte hat. Als Beispiel für eine redundante Kontaktsituation kann man sich vier Eckpunkte einer (ebenen) Fläche vorstellen, die alle mit einer anderen (ebenen) Fläche in Kontakt sind. Dadurch ist das Gleichungssystem unterbestimmt. Dann sind auch die während der Iteration auftretenden linearen Gleichungssysteme nicht eindeutig lösbar. Daher sollte man zum Lösen dieser linearen Systeme ein Verfahren wie z.B. Singulärwertzerlegung benutzen, das mit solchen Redundanzen fertig wird.

Bei dieser Vorgehensweise wird davon ausgegangen, daß die Kontakte *bilateral* sind. Bei bilateralen Kontakten ist nicht nur das sich gegenseitige Durchdringen, sondern auch das sich voneinander Lösen der Objekte verboten, wie es z.B. bei gelenkig verbundenen Objekten erwünscht ist. Das kommt daher, daß die Kontaktkräfte stets so bestimmt werden, daß die Kontaktdistanzen in den bestehenden Kontakten gleich Null werden, egal, ob die entsprechenden Objektteile sich aufeinander zu oder voneinander weg bewegen wollen. Wir wollen aber *unilaterale* Kontakte simulieren. Die Objekte sollen sich zwar nicht durchdringen, wohl aber voneinander lösen können. Dies können wir dadurch erreichen, daß wir in dem in Abschnitt 5.2 vorgestellten Algorithmus zur Bewegungssimulation am Ende nicht nur diejenigen Kontakte lösen, in denen sich die Objektteile nicht mehr berühren (weil z.B. die Ecke aus der Fläche „herausgewandert“ ist), sondern auch diejenigen Kontakte, für die eine nicht positive Kontaktkraft berechnet wurde. Wir lösen einen Kontakt also auch, wenn entweder gar keine Kraft notwendig war, um eine Durchdringung zu verhindern, oder wenn eine anziehende Kraft notwendig war, um den Kontakt aufrecht zu erhalten. Ein Problem bleibt dabei aber bestehen: Da das Ablösen der Kontakte erst am Ende des Simulationsschrittes geschieht, wirken sich negative Kontaktkräfte schon auf die Objekte aus, bevor die entsprechenden Kontakte gelöst werden. Dadurch hat man für die Zeit eines Bewegungsschrittes anziehende Kräfte. Dieses Problem wird zunehmend vernachlässigbar, je kleiner man die Schrittweite wählt. Dies führt aber dazu, daß für eine Bewegung viel mehr Einzelschritte simuliert werden müssen, was einen großen Aufwand an Rechenzeit nach sich zieht. Offensichtlich würde das Verfahren das Gewünschte tun, wenn man schon vor der Berechnung der Kontaktkräfte die relevanten Kontakte kennen würde. Für die richtige Auswahl der Kontakte ist aber bislang kein besseres Verfahren bekannt, als alle 2^K Möglichkeiten durchzuprobieren. Es ist klar, daß wir einen solchen Aufwand nicht in Kauf nehmen wollen.

5.3.2 Formulierung als nichtlineares Komplementaritätsproblem (NCP-Verfahren)

Wir wollen nun ein Verfahren vorstellen, bei dem der Effekt, daß die Objekte aneinander „kleben“, nicht auftritt. Dazu müssen wir verhindern, daß negative Kontaktkräfte berechnet werden. Wir fordern also, daß der gesuchte Vektor $\boldsymbol{f}$ der Kontaktkräfte komponentenweise größer oder gleich Null ist. Dafür schreiben wir $\boldsymbol{f} \geq \mathbf{0}$. Dann müssen wir auch zulassen, daß die Kontaktdistanzen positiv werden, da die Objekte nicht mehr daran gehindert werden, sich voneinander zu lösen. Das heißt, wir müssen für die Kontaktdistanzen $w_k(\boldsymbol{f}) \geq 0$ fordern. Eine weitere Bedingung, die wir stellen, ist, daß für alle $k = 1, \ldots, K$ nicht f_k und $w_k(\boldsymbol{f})$ beide von Null verschieden sind. Diese Bedingung begründet sich dadurch, daß eine Kontaktkraft f_k nur dann von Null verschieden sein sollte, wenn sich die entsprechenden Objektteile auch tatsächlich berühren. Wir ersetzen also das Gleichungssystem (5.5) durch das nichtlineare Komplementaritätsproblem (NCP)

$$\boldsymbol{w}(\boldsymbol{f}) \geq \mathbf{0}, \quad \boldsymbol{f} \geq \mathbf{0}, \quad \boldsymbol{f}^T \boldsymbol{w}(\boldsymbol{f}) = 0. \tag{5.6}$$

Dieser Ansatz ist eine Weiterentwicklung des in [Bar94] vorgestellten Verfahrens. Dort werden nicht die Kontaktdistanzen, sondern die relativen Kontaktbeschleunigungen benutzt, was den Vorteil hat, daß man ein *lineares* Komplementaritätsproblem (LCP) anstatt eines NCP erhält. Ein großer Nachteil dabei ist aber, daß dann die Bedingung $\boldsymbol{w}(\boldsymbol{f}) \geq \mathbf{0}$ aufgrund der Zeitdiskretisierung nicht exakt eingehalten wird. Diese Fehler akkumulieren sich im Verlaufe der Simulation.

In [Kan96] wird ein Verfahren vorgeschlagen, mit dessen Hilfe man das NCP (5.6) lösen kann. Dazu betrachten wir als Hilfsfunktion die sog. *Fischer Funktion* $\varphi : \mathbb{R}^2 \to \mathbb{R}$ mit $\varphi(a,b) = \sqrt{a^2+b^2} - a - b$. Offensichtlich hat φ die folgende Eigenschaft, die wir als *NCP-Eigenschaft* bezeichnen:

$$\varphi(a,b) = 0 \quad \Longleftrightarrow \quad a \geq 0, \quad b \geq 0, \quad ab = 0.$$

In [Kan96] werden weitere Funktionen φ erwähnt, die die NCP-Eigenschaft haben, wie z.B.

$$\begin{aligned}
\varphi(a,b) &= -ab + \frac{1}{2}\min^2\{0, a+b\}, \\
\varphi(a,b) &= -ab + \min^2\{0, a\} + \min^2\{0, b\}, \\
\varphi(a,b) &= \min\{a, b\}.
\end{aligned}$$

Mit Hilfe der Fischer Funktion definieren wir $\boldsymbol{F} : \mathbb{R}^{2K} \to \mathbb{R}^{2K}$ durch

$$\boldsymbol{F}(\boldsymbol{f}, \boldsymbol{\xi}) = \begin{pmatrix} \boldsymbol{w}(\boldsymbol{f}) - \boldsymbol{\xi} \\ \boldsymbol{\Phi}(\boldsymbol{f}, \boldsymbol{\xi}) \end{pmatrix}, \tag{5.7}$$

wobei $\boldsymbol{f}, \boldsymbol{\xi} \in \mathbb{R}^K$ und $\boldsymbol{\Phi}(\boldsymbol{f}, \boldsymbol{\xi}) = [\varphi(f_1, \xi_1), \ldots, \varphi(f_K, \xi_K)]^T$. Dann gilt:

Lemma 5.1. *Sei $\boldsymbol{f} \in \mathbb{R}^K$. Es gibt genau dann ein $\boldsymbol{\xi} \in \mathbb{R}^K$ mit $\boldsymbol{F}(\boldsymbol{f}, \boldsymbol{\xi}) = \boldsymbol{0}$, wenn $\boldsymbol{f}$ das NCP (5.6) löst.*

Beweis. Seien zunächst $\boldsymbol{f}, \boldsymbol{\xi} \in \mathbb{R}^K$ so daß $\boldsymbol{F}(\boldsymbol{f}, \boldsymbol{\xi}) = \boldsymbol{0}$ gilt. Dann gilt nach (5.7) $\boldsymbol{\Phi}(\boldsymbol{f}, \boldsymbol{\xi}) = \boldsymbol{0}$. Nach Definition von $\boldsymbol{\Phi}$, und weil die Fischer Funktion die NCP-Eigenschaft hat, gilt dann

$$\boldsymbol{\xi} \geq \boldsymbol{0}, \quad \boldsymbol{f} \geq \boldsymbol{0}, \quad \boldsymbol{f}^T \boldsymbol{\xi} = \boldsymbol{0}.$$

Ferner gilt nach (5.7) $\boldsymbol{w}(\boldsymbol{f}) = \boldsymbol{\xi}$, woraus die Behauptung folgt.
Umgekehrt erfülle nun $\boldsymbol{f}$ das NCP (5.6). Dann setzen wir $\boldsymbol{\xi} := \boldsymbol{w}(\boldsymbol{f})$. Dann ist offensichtlich $\boldsymbol{F}(\boldsymbol{f}, \boldsymbol{\xi}) = \boldsymbol{0}$. □

Damit haben wir die Lösung des Komplementaritätsproblemes auf die Lösung eines nichtlinearen Gleichungssystemes vom Grad $2K$ zurückgeführt. Dies können wir z.B. wie in (5.3.1) mit Hilfe des Newton-Raphson-Verfahrens lösen. In jeder Iteration des Newton-Raphson-Verfahrens wird dann ein lineares Gleichungssystem vom Grad $2K$ gelöst. Da aber die zusätzlich eingeführten Hilfsvariablen $\xi_1, \ldots, \xi_K$ keine unabhängigen Variablen sind, können wir dieses Verfahren etwas modifizieren, so daß nur ein lineares Gleichungssystem vom Grad K in jedem Iterationsschritt gelöst werden muß. Wie zuvor beginnen wir mit Startwerten $\boldsymbol{f}^{(0)}, \boldsymbol{\xi}^{(0)}$ und setzen für $j \geq 0$ an:

$$\frac{d\boldsymbol{F}(\boldsymbol{f}^{(j)}, \boldsymbol{\xi}^{(j)})}{d(\boldsymbol{f}, \boldsymbol{\xi})} \cdot \begin{pmatrix} \Delta \boldsymbol{f}^{(j)} \\ \Delta \boldsymbol{\xi}^{(j)} \end{pmatrix} = -\boldsymbol{F}(\boldsymbol{f}^{(j)}, \boldsymbol{\xi}^{(j)}). \tag{5.8}$$

Dabei gilt für die Jakobi-Matrix

$$\frac{d\boldsymbol{F}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)})}{d(\boldsymbol{f},\boldsymbol{\xi})} = \begin{pmatrix} \frac{d\boldsymbol{w}}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)}) & -\mathbf{I} \\ \frac{d\Phi}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) & \frac{d\Phi}{d\boldsymbol{\xi}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) \end{pmatrix}.$$

Setzt man dies und (5.7) in (5.8) ein und multipliziert aus, so erhält man die beiden Gleichungen

$$\frac{d\boldsymbol{w}}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)}) \cdot \Delta\boldsymbol{f}^{(j)} - \Delta\boldsymbol{\xi}^{(j)} = \boldsymbol{\xi}^{(j)} - \boldsymbol{w}(\boldsymbol{f}^{(j)}) \tag{5.9}$$

$$\frac{d\Phi}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) \cdot \Delta\boldsymbol{f}^{(j)} + \frac{d\Phi}{d\boldsymbol{\xi}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) \cdot \Delta\boldsymbol{\xi}^{(j)} = -\Phi(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}). \tag{5.10}$$

Stellen wir (5.9) nach $\Delta\boldsymbol{\xi}^{(j)}$ um und setzen in (5.10) ein, so erhalten wir

$$\begin{aligned} &\left(\frac{d\Phi}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) + \frac{d\Phi}{d\boldsymbol{\xi}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) \cdot \frac{d\boldsymbol{w}}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)})\right) \cdot \Delta\boldsymbol{f}^{(j)} \\ = \quad & -\Phi(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) - \frac{d\Phi}{d\boldsymbol{\xi}}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)}) \cdot (\boldsymbol{w}(\boldsymbol{f}^{(j)}) - \boldsymbol{\xi}^{(j)}). \end{aligned} \tag{5.11}$$

Wir lösen dieses lineare Gleichungssystem vom Grad K um $\Delta\boldsymbol{f}^{(j)}$ zu bestimmen und errechnen $\Delta\boldsymbol{\xi}^{(j)}$ mittels (5.9):

$$\Delta\boldsymbol{\xi}^{(j)} = \frac{d\boldsymbol{w}}{d\boldsymbol{f}}(\boldsymbol{f}^{(j)}) \cdot \Delta\boldsymbol{f}^{(j)} + \boldsymbol{w}(\boldsymbol{f}^{(j)}) - \boldsymbol{\xi}^{(j)}.$$

Danach setzen wir

$$\begin{aligned} \boldsymbol{f}^{(j+1)} &= \boldsymbol{f}^{(j)} + \Delta\boldsymbol{f}^{(j)} \text{ und} \\ \boldsymbol{\xi}^{(j+1)} &= \boldsymbol{\xi}^{(j)} + \Delta\boldsymbol{\xi}^{(j)}. \end{aligned}$$

Dies iterieren wir, bis $|\boldsymbol{F}(\boldsymbol{f}^{(j)},\boldsymbol{\xi}^{(j)})|$ unter eine vorgegebene Schranke fällt. Der zuletzt berechnete Lösungsvektor $\boldsymbol{f}^{(j)}$ ist dann unsere gesuchte Lösung. Auch hier bieten sich als Startwerte für $\boldsymbol{f}^{(0)}$ und $\boldsymbol{\xi}^{(0)}$ die Lösungswerte aus dem vorangegangenen Simulationsschritt an.

Bei der Berechnung der Systemmatrix und der rechten Seite des Gleichungssystems (5.11) können wir die einfache Struktur der Ableitungen

$$\begin{aligned} \frac{d\Phi}{d\boldsymbol{f}}(\boldsymbol{f},\boldsymbol{\xi}) &= \begin{pmatrix} \frac{f_1}{\sqrt{f_1^2+\xi_1^2}} - 1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{f_K}{\sqrt{f_K^2+\xi_K^2}} - 1 \end{pmatrix} \text{ und} \\ \frac{d\Phi}{d\boldsymbol{\xi}}(\boldsymbol{f},\boldsymbol{\xi}) &= \begin{pmatrix} \frac{\xi_1}{\sqrt{f_1^2+\xi_1^2}} - 1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{\xi_K}{\sqrt{f_K^2+\xi_K^2}} - 1 \end{pmatrix} \end{aligned}$$

ausnutzen.

5.3.3 Das LCP-Verfahren

Die in den beiden letzten Abschnitten vorgestellten Verfahren unterscheiden sich darin, daß die Kontakte einmal als bilaterale und einmal als unilaterale Kontakte aufgefaßt werden. Eine Gemeinsamkeit beider Verfahren ist, daß die Berechnung der Kontaktkräfte in beiden Fällen auf eine Folge von linearen Gleichungssystemen zurückgeführt wird. Das Verfahren, das wir jetzt beschreiben wollen, ist eine Alternative zum NCP-Verfahren. Es geht ebenfalls von unilateralen Kontakten aus. Der Unterschied zu den beiden vorigen Verfahren ist, daß jetzt keine Folge von linearen Gleichungssystemen, sondern eine Folge von linearen Komplementaritätsproblemen (LCP) generiert wird. Das Verfahren wird in ähnlicher Form in [SS98a] beschrieben und ist eine Präzisierung einer in [ST95] beschriebenen Idee.

Zunächst müssen wir noch etwas zusätzliche Notation einführen. Wir haben schon den Konfigurationsvektor $\boldsymbol{s} \in \mathbb{R}^{7N}$ gesehen:

$$\boldsymbol{s} = [\boldsymbol{c}_1^T, \mathbf{q}_1^T, \ldots, \boldsymbol{c}_N^T, \mathbf{q}_N^T]^T.$$

Wir kennen auch schon den Vektor $\boldsymbol{f} \in \mathbb{R}^K$ der Kontaktkräfte

$$\boldsymbol{f} = [f_1, \ldots, f_K]^T.$$

Wir benötigen noch den verallgemeinerten Geschwindigkeitsvektor $\boldsymbol{u} \in \mathbb{R}^{6N}$

$$\boldsymbol{u} = [\boldsymbol{v}_1^T, \boldsymbol{\omega}_1^T, \ldots, \boldsymbol{v}_N^T, \boldsymbol{\omega}_N^T]^T$$

und die Matrix $\mathbf{S} \in \mathbb{R}^{7N \times 6N}$

$$\mathbf{S} = \begin{bmatrix} \mathbf{E} & & & & \mathbf{0} \\ & \mathbf{Q}_1 & & & \\ & & \ddots & & \\ & & & \mathbf{E} & \\ \mathbf{0} & & & & \mathbf{Q}_N \end{bmatrix}.$$

Dabei ist $\mathbf{E}$ die (3×3)-Einheitsmatrix, und wenn für $1 \leq i \leq N$ das Quaternion $\mathbf{q}_i$ die Form $\mathbf{q}_i = [q_{i0}, \ldots, q_{i3}]$ hat, so ist die Matrix $\mathbf{Q}_i \in \mathbb{R}^{4\times 3}$ definiert durch

$$\mathbf{Q}_i = \frac{1}{2} \begin{bmatrix} -q_{i1} & -q_{i2} & -q_{i3} \\ q_{i0} & q_{i3} & -q_{i1} \\ -q_{i3} & q_{i0} & q_{i2} \\ q_{i2} & -q_{i1} & q_{i0} \end{bmatrix}.$$

Wie man leicht nachrechnet, gilt damit für einen Vektor $\boldsymbol{\omega} \in \mathbb{R}^3$ die Beziehung $\mathbf{Q}_i\boldsymbol{\omega} = \frac{1}{2}\boldsymbol{\omega}\mathbf{q}_i$. Daher können wir mit Hilfe dieser Notation die kontinuierlichen Bewegungsgleichungen (3.27) und (3.28) sowie die diskretisierten Bewegungsgleichungen (3.34) und (3.35) für alle Objekte gleichzeitig formulieren durch

$$\begin{aligned} \dot{\boldsymbol{s}} &= \mathbf{S}\boldsymbol{u}, \qquad \text{sowie} \\ \boldsymbol{s}^{t+\Delta t} &= \boldsymbol{s}^t + \Delta t \mathbf{S}^t \boldsymbol{u}^{t+\Delta t}. \end{aligned} \tag{5.12}$$

Wir wollen auch die Bewegungsgleichungen (5.3) und (5.4) für die Geschwindigkeiten und Winkelgeschwindigkeiten mittels einer geeigneten Notation für alle Objekte simultan formulieren können. Dazu müssen wir noch die verallgemeinerte Massenmatrix $\mathbf{M} \in \mathbb{R}^{6N \times 6N}$ definieren:

$$\mathbf{M} = \begin{bmatrix} m_1\mathbf{E} & & & & \mathbf{0} & \\ & \mathbf{I}_1 & & & & \\ & & \ddots & & & \\ & & & & m_N\mathbf{E} & \\ & \mathbf{0} & & & & \mathbf{I}_N \end{bmatrix}.$$

Weiterhin benötigen wir die Matrix $\mathbf{N} \in \mathbb{R}^{3K \times K}$ der Kontaktnormalen:

$$\mathbf{N} = \begin{bmatrix} \boldsymbol{n}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{n}_K \end{bmatrix}.$$

Dann gilt die Beziehung

$$\mathbf{N}\boldsymbol{f} = [f_1\boldsymbol{n}_1^T, \ldots, f_K\boldsymbol{n}_K^T]^T.$$

Wir wollen nun noch eine Matrix $\mathbf{J} \in \mathbb{R}^{6N \times 3K}$ definieren, so daß die Gleichung

$$\mathbf{J}\mathbf{N}\boldsymbol{f} = \begin{bmatrix} \sum\limits_{k \in \{k|i_k=1\}} f_k\boldsymbol{n}_k & - & \sum\limits_{k \in \{k|j_k=1\}} f_k\boldsymbol{n}_k \\ \sum\limits_{k \in \{k|i_k=1\}} \boldsymbol{r}_{k1} \times f_k\boldsymbol{n}_k & - & \sum\limits_{k \in \{k|j_k=1\}} \boldsymbol{r}_{k1} \times f_k\boldsymbol{n}_k \\ & \vdots & \\ \sum\limits_{k \in \{k|i_k=N\}} f_k\boldsymbol{n}_k & - & \sum\limits_{k \in \{k|j_k=N\}} f_k\boldsymbol{n}_k \\ \sum\limits_{k \in \{k|i_k=N\}} \boldsymbol{r}_{kN} \times f_k\boldsymbol{n}_k & - & \sum\limits_{k \in \{k|j_k=N\}} \boldsymbol{r}_{kN} \times f_k\boldsymbol{n}_k \end{bmatrix}$$

gilt. Man sieht, daß dazu $\mathbf{J}$ die folgende Gestalt haben muß:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \ldots & \mathbf{J}_{1K} \\ \mathbf{J}'_{11} & \ldots & \mathbf{J}'_{1K} \\ \vdots & & \vdots \\ \mathbf{J}_{N1} & \ldots & \mathbf{J}_{NK} \\ \mathbf{J}'_{N1} & \ldots & \mathbf{J}'_{NK} \end{bmatrix}.$$

Dabei sind $\mathbf{J}_{ik}, \mathbf{J}'_{ik} \in \mathbb{R}^{3\times 3}$ für $1 \le i \le N$ und $1 \le k \le K$ definiert durch

$$\mathbf{J}_{ik} = \begin{cases} \mathbf{E} & \text{falls } i_k = i, \\ -\mathbf{E} & \text{falls } j_k = i, \\ \mathbf{0} & \text{sonst,} \end{cases}$$

$$\mathbf{J}'_{ik} = \begin{cases} \boldsymbol{r}_{ki}^{\times} & \text{falls } i_k = i, \\ -\boldsymbol{r}_{ki}^{\times} & \text{falls } j_k = i, \\ \mathbf{0} & \text{sonst.} \end{cases}$$

Zum Beispiel sieht die Matrix $\mathbf{J}$ für die in Abbildung 5.2 dargestellte Kontaktsituation so aus:

$$\begin{bmatrix} \mathbf{E} & \mathbf{0} & \mathbf{E} & \mathbf{0} \\ \boldsymbol{r}_{11}^{\times} & \mathbf{0} & \boldsymbol{r}_{31}^{\times} & \mathbf{0} \\ -\mathbf{E} & -\mathbf{E} & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{r}_{12}^{\times} & -\boldsymbol{r}_{22}^{\times} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} & -\mathbf{E} & -\mathbf{E} \\ \mathbf{0} & \boldsymbol{r}_{23}^{\times} & -\boldsymbol{r}_{33}^{\times} & -\boldsymbol{r}_{43}^{\times} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{r}_{44}^{\times} \end{bmatrix}.$$

Definieren wir nun noch den Vektor $\boldsymbol{f}_e \in \mathbb{R}^{6N}$ durch

$$\boldsymbol{f}_e = [\mathbf{0}^T, (-\boldsymbol{\omega}_1 \times \mathbf{I}_1\boldsymbol{\omega}_1)^T, \ldots, \mathbf{0}^T, (-\boldsymbol{\omega}_N \times \mathbf{I}_N\boldsymbol{\omega}_N)^T]^T,$$

dann können wir die Gleichungen (5.3) und (5.4) für die Geschwindigkeiten für alle Objekte simultan formulieren als

$$\boldsymbol{u}^{t+\Delta t} = \boldsymbol{u}^t + \Delta t \mathbf{M}^{-1}(\mathbf{J}\mathbf{N}\boldsymbol{f} + \boldsymbol{f}_e^t). \tag{5.13}$$

Um nun eine Bedingung für die Kontaktkräfte herzuleiten, betrachten wir die Normalkomponente der relativen Kontaktgeschwindigkeit im k-ten Kontakt zum Zeitpunkt $t + \Delta t$. Diese ist gegeben durch

$$\mu_k = \boldsymbol{n}_k^T(\boldsymbol{v}_{i_k}^{t+\Delta t} + \boldsymbol{\omega}_{i_k}^{t+\Delta t} \times \boldsymbol{r}_{ki_k}) - \boldsymbol{n}_k^T(\boldsymbol{v}_{j_k}^{t+\Delta t} + \boldsymbol{\omega}_{j_k}^{t+\Delta t} \times \boldsymbol{r}_{kj_k}).$$

Mit unserer oben eingeführten Notation ist dies für alle Kontakte gleichzeitig formuliert

$$\boldsymbol{\mu} = \mathbf{N}^T\mathbf{J}^T\boldsymbol{u}^{t+\Delta t}.$$

$\mu_k > 0$ bedeutet dann, daß die Distanz im k-ten Kontakt größer wird, und $\mu_k < 0$ bedeutet, daß die Distanz im k-ten Kontakt kleiner wird. μ_k darf also nicht beliebig klein werden, da die Kontaktdistanz im k-ten Kontakt sonst negativ wird. Es gibt also ein ν_k, so daß $\mu_k \ge \nu_k$ gelten muß. Für alle Kontakte gleichzeitig heißt das, daß ein $\boldsymbol{\nu} \in \mathbb{R}^K$ existiert mit

$$\mathbf{N}^T\mathbf{J}^T\boldsymbol{u}^{t+\Delta t} \ge \boldsymbol{\nu}. \tag{5.14}$$

Setzen wir hierin (5.13) ein und formen um, so erhalten wir

$$\Delta t \mathbf{N}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{N} \boldsymbol{f} + \mathbf{N}^T \mathbf{J}^T (\boldsymbol{u}^t + \Delta t \mathbf{M}^{-1} \boldsymbol{f}_e) - \boldsymbol{\nu} \geq \mathbf{0}.$$

Dies schreiben wir in der Form

$$\mathbf{A}\boldsymbol{f} + \boldsymbol{b} \geq \mathbf{0}$$

mit

$$\begin{aligned} \mathbf{A} &= \Delta t \mathbf{N}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{N} \quad \text{und} \\ \boldsymbol{b} &= \mathbf{N}^T \mathbf{J}^T (\boldsymbol{u}^t + \Delta t \mathbf{M}^{-1} \boldsymbol{f}_e) - \boldsymbol{\nu}. \end{aligned}$$

Wenn wir nun wüßten, wie wir $\boldsymbol{\nu}$ unabhängig von $\boldsymbol{f}$ wählen müssen, so daß die geometrische Bedingung $\boldsymbol{w}(\boldsymbol{f}) \geq 0$ erfüllt wird, so könnten wir das lineare Komplementaritätsproblem (LCP)

$$\mathbf{A}\boldsymbol{f} + \boldsymbol{b} \geq \mathbf{0}, \quad \boldsymbol{f} \geq \mathbf{0}, \quad \boldsymbol{f}^T(\mathbf{A}\boldsymbol{f} + \boldsymbol{b}) = 0$$

lösen (Wie wir ein solches LCP lösen können, sehen wir weiter unten). Schaut man sich (5.14) an, so ist offensichtlich, daß diese geometrische Bedingung erfüllt wäre, wenn wir $\boldsymbol{\nu} = -\frac{\boldsymbol{w}(\boldsymbol{f})}{\Delta t} + \mathbf{N}^T \mathbf{J}^T \boldsymbol{u}^{t+\Delta t}$ wählen würden. Dies ist aber nicht unabhängig von $\boldsymbol{f}$, da sowohl $\boldsymbol{w}(\boldsymbol{f})$ als auch $\boldsymbol{u}^{t+\Delta t}$ von $\boldsymbol{f}$ abhängt. Könnten wir aber $\boldsymbol{\nu}$ so wählen, dann wäre $\mathbf{A}\boldsymbol{f} + \boldsymbol{b} = \frac{\boldsymbol{w}(\boldsymbol{f})}{\Delta t}$, und die Lösung $\boldsymbol{f}$ des LCP wäre auch eine Lösung des im vorigen Abschnitt benutzten NCP (5.6). Aus diesen Überlegungen ergibt sich die folgende Fixpunktiteration:

Wir beginnen mit einem Startvektor $\boldsymbol{f}^{(0)}$. Für $j \geq 0$ berechnen wir dann

$$\begin{aligned} \boldsymbol{u}^{(j)} &= \boldsymbol{u}^t + \Delta t \mathbf{M}^{-1} (\mathbf{J}\mathbf{N}\boldsymbol{f}^{(j)} + \boldsymbol{f}_e^t), \\ \boldsymbol{\nu}^{(j)} &= -\frac{\boldsymbol{w}(\boldsymbol{f}^{(j)})}{\Delta t} + \mathbf{N}^T \mathbf{J}^T \boldsymbol{u}^{(j)}, \\ \boldsymbol{b}^{(j)} &= \mathbf{N}^T \mathbf{J}^T (\boldsymbol{u}^t + \Delta t \mathbf{M}^{-1} \boldsymbol{f}_e) - \boldsymbol{\nu}^{(j)} \end{aligned}$$

und lösen das LCP

$$\mathbf{A}\boldsymbol{f}^{(j+1)} + \boldsymbol{b}^{(j)} \geq \mathbf{0}, \quad \boldsymbol{f}^{(j+1)} \geq \mathbf{0}, \quad (\mathbf{A}\boldsymbol{f}^{(j+1)} + \boldsymbol{b}^{(j)})^T \boldsymbol{f}^{(j+1)} = 0. \tag{5.15}$$

Dies iterieren wir solange, bis $|\boldsymbol{f}^{(j+1)} - \boldsymbol{f}^{(j)}|$ unter eine vorgegebene Schranke fällt. Der zuletzt berechnete Lösungsvektor $\boldsymbol{f}^{(j+1)}$ ist dann unsere gesuchte Lösung, denn dann ist

$$\begin{aligned} \mathbf{A}\boldsymbol{f}^{(j+1)} + \boldsymbol{b}^{(j)} &= \mathbf{N}^T \mathbf{J}^T (\boldsymbol{u}^t + \Delta t \mathbf{M}^{-1} (\mathbf{J}\mathbf{N}\boldsymbol{f}^{(j+1)} + \boldsymbol{f}_e^t)) + \frac{\boldsymbol{w}(\boldsymbol{f}^{(j)})}{\Delta t} - \mathbf{N}^T \mathbf{J}^T \boldsymbol{u}^{(j)} \\ &= \mathbf{N}^T \mathbf{J}^T \boldsymbol{u}^{(j+1)} + \frac{\boldsymbol{w}(\boldsymbol{f}^{(j)})}{\Delta t} - \mathbf{N}^T \mathbf{J}^T \boldsymbol{u}^{(j)} \\ &= \frac{\boldsymbol{w}(\boldsymbol{f})}{\Delta t} + \mathbf{N}^T \mathbf{J}^T (\boldsymbol{u}^{(j+1)} - \boldsymbol{u}^{(j)}). \end{aligned}$$

$\boldsymbol{u}^{(j+1)}$ und $\boldsymbol{u}^{(j)}$ sind gleich, wenn $\boldsymbol{f}^{(j+1)}$ und $\boldsymbol{f}^{(j)}$ gleich sind, was am Ende der Iteration bis auf einen winzigen Fehler gilt.

Wir haben gesehen, daß jede Lösung dieses Verfahrens auch eine Lösung des NCP-Verfahrens ist. Es ist nicht schwer zu sehen, daß umgekehrt auch jede Lösung des NCP-Verfahrens eine Lösung dieses Verfahrens ist. Damit kann dieses Verfahren alternativ zum NCP-Verfahren benutzt werden.

Um die Beschreibung des LCP-Verfahrens zu vervollständigen, müssen wir noch sagen, wie man ein LCP der Form (5.15) lösen kann. Dazu beobachtet man, daß unsere Matrix $\mathbf{A} = \Delta t \mathbf{N}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{N}$ symmetrisch und positiv semidefinit ist: Die Matrizen $m_i \mathbf{E}$ haben diese Eigenschaften. Ebenso die Trägheitsmatrizen $\mathbf{I}_i$, wie man aus Gleichung (3.17) ableitet. Damit hat auch $\mathbf{M}$ diese Eigenschaften und ebenso $\mathbf{M}^{-1}$. Nun ist $\mathbf{A} = \Delta t (\mathbf{JN})^T \mathbf{M}^{-1} (\mathbf{JN})$. Daher hat auch $\mathbf{A}$ diese Eigenschaften. Ein LCP der Form

$$\mathbf{A}\boldsymbol{x} + \boldsymbol{b} \geq \mathbf{0}, \quad \boldsymbol{x} \geq \mathbf{0}, \quad \boldsymbol{x}^T (\mathbf{A}\boldsymbol{x} + \boldsymbol{b}) = 0$$

kann mit dem Lemke-Algorithmus (siehe [CPS92]) gelöst werden, wenn $\mathbf{A}$ symmetrisch und positiv definit ist. Treten keine redundanten Kontakte auf, sind diese Voraussetzungen sind bei unserer Matrix $\mathbf{A}$ erfüllt.

5.4 Bemerkung zu den physikalischen Annahmen

Nachdem wir nun Verfahren für die Berechnung von Kontaktkräften beschrieben haben, wollen wir kurz sagen, welche physikalischen Annahmen man bei der Verwendung dieser Verfahren macht. Zwei wichtige vereinfachende Annahmen haben wir schon erwähnt:

- Die Objekte sind starr, d.h. nicht verformbar.
- Es findet keine Reibung zwischen Objekten statt.

Diese Annahmen waren wichtig für die Herleitung der Verfahren. Eine weitere, nicht offensichtliche vereinfachende Annahme ist, daß alle Stöße, die zwischen Objekten auftreten, rein plastisch sind. Dies sieht man folgendermaßen. Zum einen haben wir natürlich auf die Einhaltung des Impulserhaltungssatzes geachtet, denn jede Kontaktkraft wirkt gleichermaßen auf beide beteiligte Objekte in jeweils entgegengesetzter Richtung. Zum anderen ist nach einem Stoß die relative Kontaktgeschwindigkeit in Richtung der Kontaktnormalen gleich Null. Das ist nicht offensichtlich. Um dies zu sehen, definieren wir zu einem Vektor $\boldsymbol{f}$ von Kontaktkräften die Funktion $\boldsymbol{\delta}_{\boldsymbol{f}} : \mathbb{R} \to \mathbb{R}^K$, die jedem Zeitpunkt $\tau \in [t + \Delta t]$ den Vektor der Kontaktdistanzen zuordnet, den man erhält, wenn die Kontaktkräfte $\boldsymbol{f}$ über das Zeitintervall $[t, \tau]$ gewirkt haben. Dann ist $\boldsymbol{\delta}_{\boldsymbol{f}}(t + \Delta t) = \boldsymbol{w}(\boldsymbol{f})$. In [ACP95] wird gezeigt, daß gilt

$$\dot{\boldsymbol{\delta}}_{\boldsymbol{f}} = \mathbf{N}^T \mathbf{J}^T \boldsymbol{u},$$

oder für den k-ten Kontakt formuliert:

$$\dot{\delta}_{k\boldsymbol{f}} = \boldsymbol{n}_k^T (\boldsymbol{v}_{i_k} + \boldsymbol{\omega}_{i_k} \times \boldsymbol{r}_{ki_k} - \boldsymbol{v}_{j_k} - \boldsymbol{\omega}_{j_k} \times \boldsymbol{r}_{kj_k}).$$

Diese intuitiv klare Aussage haben wir eigentlich schon benutzt, als wir bei der Begründung der Ungleichung (5.14) anhand des Vorzeichens von μ_k entschieden haben, ob die Kontaktdistanz größer oder kleiner wird.
Wir müssen jetzt nur zeigen, daß $\dot{\delta}_{k\boldsymbol{f}}$ bei einem Stoß gleich Null wird. Dazu unterscheiden wir, ob κ_k ein alter oder ein neuer Kontakt ist.

1. κ_k ist ein alter Kontakt.
   Dann ist $\delta_{k\boldsymbol{f}}(t) = 0$, weil die in Abschnitt 5.1 erwähnte Invariante gilt. Liegt ein Stoß vor, ist die Kontaktkraft $f_k \neq 0$. Dann ist $w_k(\boldsymbol{f}) = 0$, da $\boldsymbol{f}$ Lösung des NCP (5.6) ist (beim ersten Verfahren gilt das immer). Dann ist
   $$\begin{aligned} \dot{\delta}_{k\boldsymbol{f}}(t+\Delta t) &\doteq \frac{\delta_{k\boldsymbol{f}}(t+\Delta t) - \delta_{k\boldsymbol{f}}(t)}{\Delta t} \\ &= \frac{w_k(\boldsymbol{f}) - \delta_{k\boldsymbol{f}}(t)}{\Delta t} \\ &= 0. \end{aligned}$$
   Die Relation „$\doteq$“ in der ersten Zeile bedeutet „in erster Näherung gleich“.

2. κ_k ist ein neuer Kontakt.
   Dann ist $\delta_{k\boldsymbol{f}}(t) \geq 0$ und wir können nicht zeigen, daß $\dot{\delta}_{k\boldsymbol{f}}(t+\Delta t) \doteq 0$ gilt. Diese Geschwindigkeit kann nämlich negativ sein, so daß die Objekte sich am Ende des Zeitintervalls berühren (damit die Invariante aus Abschnitt 5.1 erfüllt ist). Allerdings ist κ_k dann im nächsten Zeitschritt ein alter Kontakt und der erste Fall tritt ein. Die Normalkomponente der relativen Kontaktgeschwindigkeit verschwindet dann erst nach einer Verzögerung von Δt.

Es wäre nun wünschenswert, daß man diese vereinfachenden Annahmen fallen lassen könnte, um eine größere Realitätsnähe zu erreichen. Was die Behandlung von Reibung angeht, sei z.B. [SS98b] erwähnt. Auch für die Simulation verformbarer Objekte gibt es Ansätze, wie z.B. die Finite-Elemente-Methode, die allerdings von einer Echtzeitfähigkeit noch weit entfernt ist. Elastische Stöße wurden vor allem bei der impulsbasierten Dynamiksimulation, z.B. in [Mir96b], behandelt.

6 Resultate

Wir wollen nun noch einige Resultate in Form von Beispielszenerien präsentieren, die mit SilVIA simuliert wurden.

Abbildung 6.1 zeigt eine Szene, in der eine Kreuzschlitzschraube mit Hilfe eines Kreuzschlitzschraubenziehers aus einer Mutter herausgedreht wird. Schraubenkopf und Schraubenzieherspitze sind nach der internationalen Norm ISO 4757 modelliert worden. Um mehr erkennen zu können, sind die Schraube und die Mutter transparent dargestellt. In Bild a) wurde der Schraubenzieher leicht verkantet in die Schraube eingesetzt. Durch Ausüben eines leichten Druckes und durch leichtes Drehen ist die Schraubenzieherspitze in die richtige Position geglitten, d.h. vollständig in den Kreuzschlitz hinein (Bild b) und c)). Durch weiteres Drehen des Schraubenziehers konnte die Schraube dann aus der Mutter herausgedreht werden (Bild d)). Diese relativ komplexe Szene bestand aus ca. 2000 Dreiecken. Dabei waren bis zu 20 gleichzeitige Kontakte keine Seltenheit. Vor allem im Bereich der Gewinde berühren sich die Schraube und die Mutter an sehr vielen Stellen gleichzeitig. Bei diesem Beispiel wurden die elementaren Kollisionserkennungsroutinen sehr häufig aufgerufen, da im Bereich der Gewinde die Hüllkörperhierarchien die Anzahl der zu testenden Flächenpaare kaum reduzieren konnten. Dieses Experiment wurde auf einem Linux-PC mit einer 266MHz-CPU durchgeführt und erforderte etwa 90 Minuten Rechenzeit. Bei einem ähnlichen Experiment, bei dem die Mutter und das Schraubengewinde weggelassen wurden, konnte die Schraubenzieherspitze interaktiv in den Kreuzschlitz eingeführt werden.

Die Szene, die in Abbildung 6.2 dargestellt ist, ist geometrisch sehr viel einfacher. Diese Abbildung soll eine mögliche Anwendung für eine Bewegungssimulation demonstrieren, nämlich das Plazieren eines Objektes. In der Szene soll das kleine Objekt interaktiv genau in die konkave Ecke des größeren Objektes plaziert werden. In jedem Bild stellen die Pfeile die zum entsprechenden Zeitpunkt berechneten Kontaktkräfte dar.

a)

b)

c)

d)

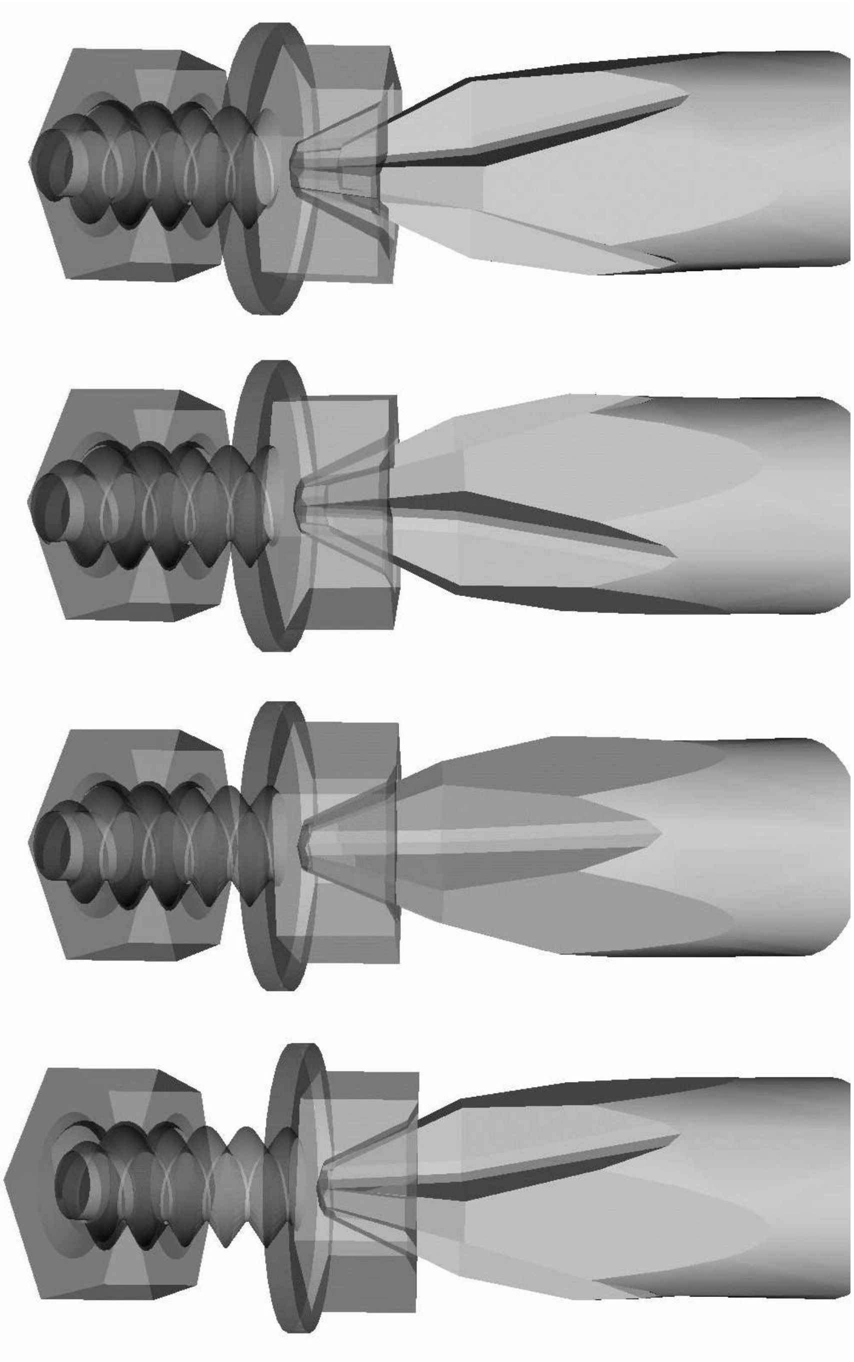

Abbildung 6.1: Beispielszene: Herausdrehen einer Kreuzschlitzschraube aus einer Mutter mit Hilfe eines Kreuzschlitzschraubenziehers

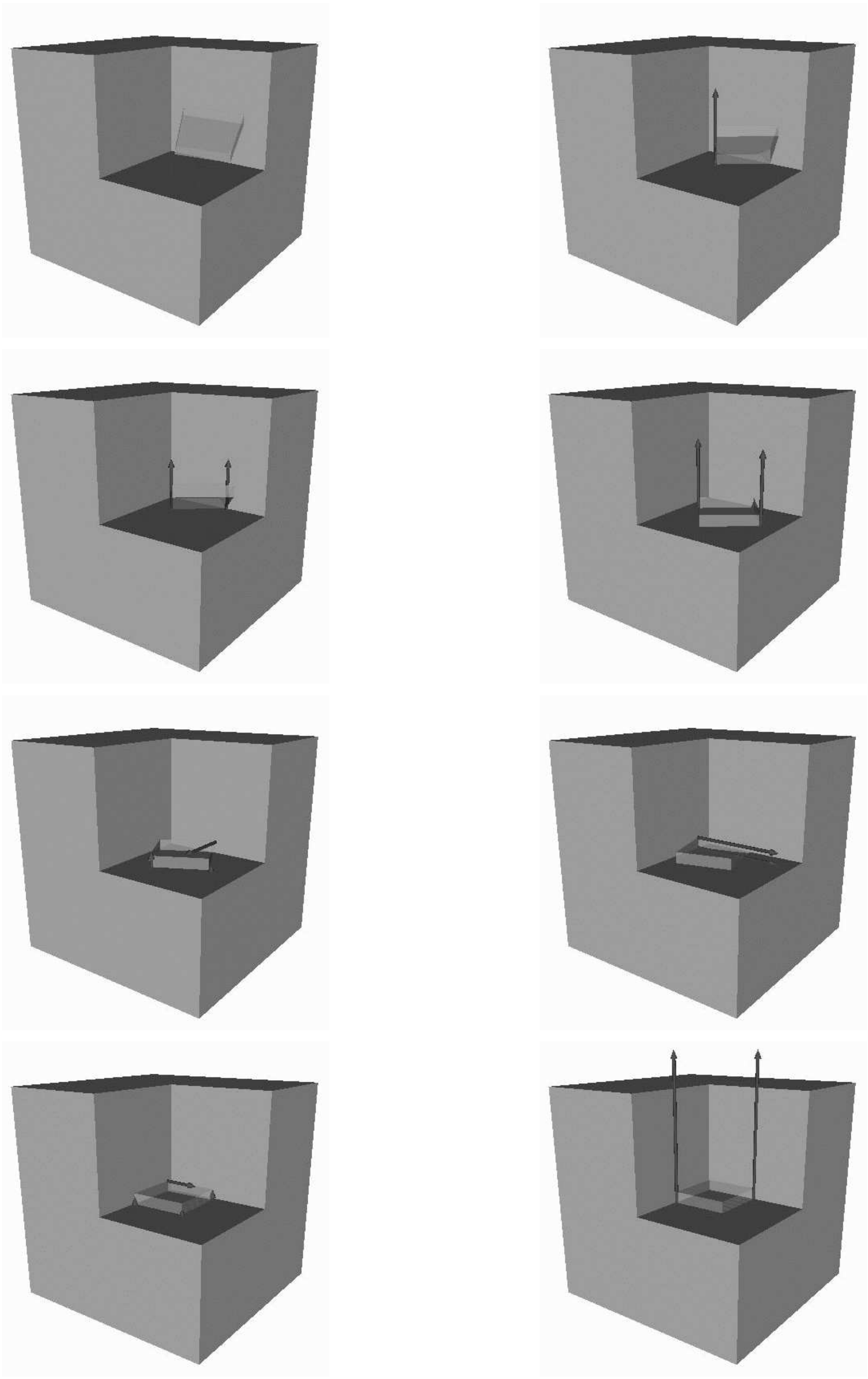

Abbildung 6.2: Beispielszene: Plazieren eines Objektes.

Literaturverzeichnis

[ACP95] M. Anitescu, J. F. Cremer, and F. A. Potra. Formulating 3d contact dynamics problems. Technical Report, Department of Mathematics, University of Iowa, 1995.

[Bar94] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH*, pages 174–203, July 1994.

[Can87] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT-press, 1987.

[Cha95] T. Chadzelek. Heuristische Bewegungsplanung mit vielen Freiheitsgraden. Master's thesis, Universität des Saarlandes, 1995.

[CPS92] R.W. Cottle, J.S. Pang, and R.E. Stone. *The Linear Complementarity Problem.* Academic Press, Inc., 1992.

[Jos97] S. Jost. Impulsbasierte Dynamik starrer Körper. Master's thesis, Universität des Saarlandes, 1997.

[Kan96] C. Kanzow. Global convergence properties of some iterative methods for linear complementarity problems. *SIAM Journal of Optimization*, 6(2):326–341, May 1996.

[Mir96a] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *J. Graphics Tools*, 1(2):31–50, 1996.

[Mir96b] B. Mirtich. *Impulse-based dynamic simulation of rigid body systems.* PhD thesis, University of California, Berkeley, 1996.

[SS98a] J. Sauer and E. Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *ACM Symposium on Virtual Reality Software and Technology*, 1998.

[SS98b] J. Sauer and E. Schömer. Real-time rigid body simulations of some classical mechanics toys. In *10th European Simulation Symposium and Exhibition, ESS'98*, 1998. to appear.

[ST95] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, submitted 1995.