CHRISTIAN LENNERZ

Impulsbasierte Dynamiksimulation starrer Körper unter Verwendung von Hüllkörperhierarchien

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

Saarbrücken 2000

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

Saarbrücken, im Februar 2000

Für meine Eltern

Inhaltsverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

1 Einleitung

Dynamical systems exhibit a great deal of beautiful structure, and dynamicists have developed a body of knowledge that can predict the behavior of many of these systems to great accuracy. The ability to know in advance how our universe or some part of it will evolve is certainly one of the most powerful skills humans ever learned. The modern computer with its power to perform computations at blinding speed and display realistic images, has greatly increased our predictive power.

– Brian Mirtich, 1996 –

Die echtzeitfähige Vorhersage der Bewegungen realer, nichtdeformierbarer Körper unter dem Einfluß externer Faktoren ist der kurzgefaßte inhaltliche Rahmen dieser Arbeit. Die Gesetzmäßigkeiten, nach denen eine solche Bewegung abläuft, werden in der *Mechanik*, einem Teilgebiet der theoretischen Physik untersucht. Die *Kinematik* behandelt dabei die bloße Beschreibung der Bewegungen, während die *Dynamik* sich mit den Ursachen von Bewegungen befaßt. Die Dynamik stellt somit die Theorien bereit, die der Simulation des kinematischen Verhaltens realer Objekte zugrundeliegt. Die mathematische Basis zur Analyse dynamischer Systeme von nichtdeformierbaren, sogenannten starren Körpern wurde in den letzten 350 Jahren von NEWTON, EULER, LAGRANGE und anderen entwickelt. Die Theorie der klassischen Mechanik geht dabei von gewissen unbewiesenen, grundlegenden Gesetzen aus, den *Newtonschen Axiomen* bzw. deren Verallgemeinerung, den *Lagrangegleichungen*. Das zentrale Ergebnis dieser Theorie ist die *quantitative* Beschreibung der Bewegung eines starren Körpers unter dem Einfluß von externen Kräften. Das Verhalten eines einfachen dynamischen Systems, wie beispielsweise die Flugbahn eines Balls unter Einwirkung von Gravitation, läßt sich dabei noch ohne technische Hilfsmittel voraussagen, ja sogar mathematisch in geschlossener Form ausdrücken. Ganz anders sieht die Situation im Fall eines Dosenstapels auf dem Jahrmarkt aus. Sollen die scheinbar chaotischen Flugbahnen der einzelnen Dosen, die durch einen gezielten Ballwurf aus der Ruhelage bewegt wurden, ohne Computerunterstützung vorausgesagt werden, so ist die Grenze der „manuellen“ Anwendbarkeit der dynamischen Theorie bereits überschritten. Ein solches Systemverhalten kann nicht mehr mathematisch in geschlossener Form beschrieben werden, sondern muß unter Zuhilfenahme von Computern schrittweise simuliert werden.

Doch selbst rechnergestützt war in den Anfängen der Dynamiksimulation das Ziel einer wirklichkeitsgetreuen Voraussage von Objektbewegungen nicht zu erreichen. Aus diesem Grund wurden Dynamiksimulationen zunächst in einem eingeschränkten Kontext studiert. Entweder untersuchte man spezielle Systeme, die besonders einfache

Simulationsberechnungen zuließen, oder man ging von teilweise dubiosen, vereinfachenden Annahmen aus, die die Aussagekraft der Simulationsdaten stark einschränkten. Mit zunehmender Rechenleistung gelang es, die Grenzen des Machbaren hinauszuschieben. So war es schließlich möglich, allgemeinere und physikalisch präzisere Verfahren einzusetzen, doch das nächste Ziel der Echtzeitfähigkeit war noch immer weit entfernt. Die Fortschritte im Rahmen der Hardwaretechnologie und vor allem die Verbesserung bestehender, sowie die Entwicklung neuer Modelle und Algorithmen hat erst in diesem Jahrzehnt die Grundlagen für die Erreichung der drei hochgesteckten Ziele *Echtzeitfähigkeit*, *allgemeine Anwendbarkeit* und *physikalische Genauigkeit* gelegt. ANDREW WITKIN, einer der Vorreiter der Dynamiksimulation, drückte diese Errungenschaft 1990 folgendermaßen aus:

> „ *The increasing availability of high-performance computers with fast 3D graphics has for the first time made it possible to perform non-trivial physical simulations – and see the results – at fully interactive speeds.*“

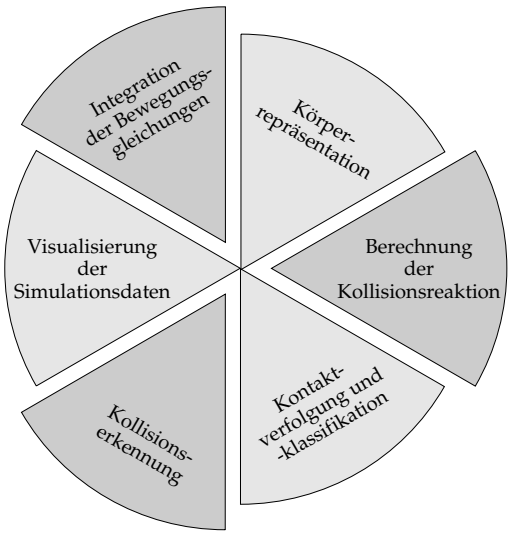
1.1 Problemstellungen und Forschungsgebiete

> *Many pieces are needed to solve the puzzle [of dynamic simulation], and a wide variety of research efforts are attacking the problem on many fronts.*
>
> – Brian Mirtich, 1996 –

Der Grund, warum wir in komplexeren dynamischen Szenarien auf die Unterstützung von Computern zur Vorhersage des qualitativen Systemverhaltens angewiesen sind, liegt nicht nur in der großen Zahl der Objekte, sondern vor allem in der Notwendigkeit, die Vielfalt ihrer Interaktionsmöglichkeiten berücksichtigen zu müssen. Dabei spielen auftretende Kontakte eine besondere Rolle, weil sie komplexe Wechselwirkungen zwischen den Kollisionspartnern induzieren und somit Unstetigkeiten in den Objektbewegungen verursachen. Da die Berechnung der Bewegung isolierter, starrer Körper unter nichtkontaktinduzierten Kräften sehr gut verstanden ist, stellt die *verläßliche Erkennung* und *physikalische korrekte Behandlung* solcher kontaktinduzierten Bewegungseinflüsse den kritischen Erfolgsfaktor einer jeden Dynamiksimulation dar.

Kollisionen verursachen Interaktionen zwischen den Objekten, die auf mikroskopischem Niveau stattfinden und somit im allgemeinen der rechnergestützten Analyse versagt bleiben. Es müssen daher *Kollisionsmodelle* entwickelt werden, die von den atomaren Wechselwirkungen abstrahieren und dennoch eine präzise Voraussage des Kontaktverhaltens erlauben. Das von uns verwendete impulsbasierte Modell der *Kollisionsreaktion* wendet dazu die Stoßgesetze im Kontaktpunkt an und gestattet auf diese Weise, die Impulsübertragung zwischen den Körpern zu erfassen. In dieser Arbeit werden wir weitere Modelle kennenlernen, die völlig andere Ansätze verfolgen. Allen gemeinsam ist jedoch, daß sie sich bezüglich des *inhärenten Trade-Offs* zwischen *Effizienz* und *physikalischer Präzision* positionieren lassen. Der von uns gewählte impulsbasierte

Abbildung 1.1 Die Zentralen Problemstellungen der Dynamiksimulation.



Ansatz von MIRTICH [Mir96b] versucht, die konkurrierenden Anforderungen bezüglich Echtzeitfähigkeit und qualitativer Korrektheit der Simulationsdaten in Einklang zu bringen. Die Entwicklung von dynamischen Kollisionsmodellen und Verfahren zur Berechnung der *Kollisionsreaktion* fällt in ein sehr aktuelles Forschungsgebiet mit dem Namen *algorithmische Mechanik* (*Computational Mechanics*).

Die zweite zentrale Problemstellung ist wie bereits angedeutet das Erkennen von Kollisionen zwischen sich bewegenden Objekten. Die Verläßlichkeit der *Kollisionserkennung* ist genauso wie die physikalische Validität der Kollisionsreaktion kritisch für die Aussagekraft der Simulation. Nichtentdeckte und somit nichtbehandelte Kollisionen verfälschen die Simulationsergebnisse nachhaltig. Im Zusammenhang mit dem Problem der Kollisionserkennung verlassen wir das Gebiet der (algorithmischen) Mechanik und wenden uns primär geometrischen Fragestellungen zu, die im Bereich der *algorithmischen Geometrie* bzw. *Robotik* anzusiedeln sind. Die besondere Herausforderung für die Kollisionserkennung im Umfeld der Dynamiksimulation ist neben der bereits angesprochenen Verläßlichkeit und Echtzeitfähigkeit die Berücksichtigung und Verwertung der Dynamikinformationen, die in verwandten Fragestellungen der algorithmischen Geometrie nicht zur Verfügung stehen. Auch die echtzeitfähige Behandlung der Kollisionserkennungsfrage erfordert Modelle und Vereinfachungen, die vor allem die Beschreibung des Körpers betreffen.

Die dritte Schlüsselkomponente der Dynamiksimulation, die jedoch nicht mehr als Forschungsgebiet bezeichnet werden kann, ist die Bewegungsberechnung durch Integration der Bewegungsgleichungen von NEWTON und EULER.

Als weitere Komponente wird häufig die *Kontaktverfolgung* und *-klassifikation* genannt. Dabei versucht man zwischen Stößen, also *vorübergehenden Kontakten*, und *per-*

manenten Kontakten wie beispielsweise Rollen und Gleiten algorithmisch zu unterscheiden, mit dem Ziel, verschiedene Kontaktformen anhand der für sie optimalen Verfahren der Kollisionsauflösung zu behandeln. Man sieht leicht ein, daß die Identifikation der vorliegenden Kontaktform keinesfalls zeitpunktbezogen erfolgen kann, sondern die in der Vergangenheit aufgelösten Kollisionen berücksichtigen muß. Eine solche Komponente ist zwar wünschenswert, jedoch nicht zwingend zur erfolgreichen Umsetzung eines Dynamiksimulationskonzeptes erforderlich.

Die Dynamiksimulation ist offensichtlich ein interdisziplinäres Gebiet, da sie neben den bereits erwähnten Bereichen wie Robotik, algorithmische Geometrie und Mechanik eine Reihe anderer Forschungsgebiete erfaßt. So stellt die *Visualisierung* großer dynamischer Systeme eine große Herausforderung an die Graphikalgorithmen und -hardware. Die *(Distributed) Virtual-Reality*-Forschung erhält durch die Kollisionsreaktionsverfahren weiteren Auftrieb, da nun die Möglichkeit besteht, haptisches Feedback an den Benutzer weiterzugeben. Die Gestaltung einer *effizienten und intelligenten Interaktion mit dem Benutzer* sowie die Weiterverarbeitung der Simulationsdaten mit dem Ziel einer *empirischen Optimierung des simulierten Systems* ist eine bedeutsame Aufgabe für die Forschung im Bereich *künstlicher Intelligenz*. Abbildung 1.1 verdeutlicht die zu lösenden Problemstellungen.

1.2 Anwendungsgebiete

In den vergangenen Jahren hat das Interesse an Echtzeitdynamiksimulationen sehr stark zugenommen. Im *Unterhaltungssektor* eröffnet sie vor allem im Rahmen der *Animationserstellung* für Film und Werbung vielfältige Möglichkeiten, die konventionellen Techniken verborgen blieben. Die Integration der Simulationsalgorithmen in *Computerspiele* erlaubt bisher unerreichbare Realitätsnähe und verspricht völlig neuen Spielespaß durch wirklichkeitsgetreue Objektbewegungen. Die Berechnung von Kräften bzw. Impulsen, die eine Durchdringung der Objekte verhindern, erweitert die Empfindung physischer Präsenz (*Immersion*) in virtuellen Welten und verbessert somit die Leistungsfähigkeit von *Virtual-Reality-Trainingsmethoden*, welche beispielsweise im Bereich der *Operationsplanung* und *-ausbildung* bereits eingesetzt werden. In der *Wissenschaft* und *Ausbildung* erlauben Dynamiksimulationen ein tieferes Verständnis komplexer dynamischer Prozesse und ersetzen somit zeit- und kostenintensive Experimente. Die Integration in *Lernsoftware* nimmt zudem einen Teil der Abstraktheit der theoretischen Mechanik aus den Schulen und Hörsälen. In der Industrie können Konzepte wie beispielsweise *Virtual Prototyping* und *präemptive Qualitätssicherung* praktisch umgesetzt werden. Diese ermöglichen es, durch effiziente Prozeßorganisation Zeit und Kosten bei hohem Qualitätsniveau einzusparen und über resultierende Wettbewerbsvorteile Marktanteile zu gewinnen.

Zwei zentrale Anwendungsgebiete wollen wir im folgenden hinsichtlich der Einsatzmöglichkeiten von Dynamiksimulationen detaillierter vorstellen. Dies ist zum einen die *Computeranimation*, da sie sich in besonderem Maße für das in dieser Arbeit entwickelte Simulationssystem empfiehlt und zum anderen das *Virtual Prototyping-*

Konzept, auf das die Simulationsbibliothek SILVIA zielt, in die die implementierten Komponenten integriert wurden.

Computeranimation

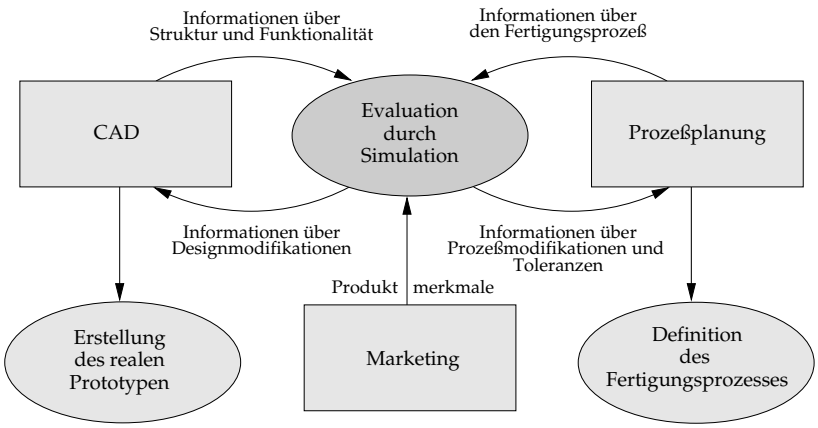
> *Most people think the word 'animation' means movement. But it doesn't. It comes from 'animus' which means 'life or to live'. Making it move is not animation, but the mechanics of it.*
>
> – Ollie Johnston and Frank Thomas, 1984 –

Bewegte Bilder besitzen die nahezu magische Fähigkeit, unsere Aufmerksamkeit anzuziehen und unsere Phantasie zu entfachen. Animationstechniken machen es möglich, aus einer Folge künstlich generierter Standbilder den Eindruck natürlicher Bewegung und Interaktion, d.h. lebendiger Wirklichkeit zu erzeugen. Eine solche Illusion ist keine leichte Aufgabe, da Menschen sehr geübt darin sind, Bewegungen und Objektwechselwirkungen zu beobachten und zu analysieren. Sie reagieren daher sensibel auf Unstimmigkeiten und entdecken schnell, wenn eine Bewegung oder Interaktion unnatürlich oder unrealistisch ist. Aus diesem Grund muß der Animationsdesigner großen Aufwand betreiben, um eine ausreichende Menge an Details des realen Verhaltens in der Animation umzusetzen. Ist die Zahl der Objekte in der Animationsszene sehr hoch, wie wir es aus den Walt Disney®-Filmen, aus Toy Story® oder Star Wars® kennen, so wird diese Arbeit schlichtweg ermüdend und überfordert vielfach in ihrer Komplexität den Animationsdesigner.

Computeranimationstechniken unterstützen daher die klassischen Verfahren bzw. lösen diese gar ab, da ihre Fähigkeiten weit über die bisher möglichen, manuell erzeugten Effekte hinausgehen. Dynamiksimulationen, die physikalischen Gesetzen folgen, sind inhärent realistisch und stellen somit ein zentrales *prozedurales Modell* zur Erzeugung glaubwürdiger Objektbewegungen und -interaktionen dar. Dies haben bereits die Pioniere der Walt Disney®-Animation, OLLIE JOHNSTON und FRANK THOMAS, erkannt, wie das obige Zitat beweist.

Konventionelle Animationstechniken können dabei auf zwei verschiedene Arten durch den Einsatz von Dynamiksimulationen unterstützt werden. Traditionell wird eine Animation erstellt, indem bestimmte *Schlüsselszenen* (*Keyframes*) der Handlung erzeugt werden. Im Rahmen des *In-betweening*-Prozesses werden anschließend die Bilder zwischen den Schlüsselszenen generiert. Diese *Keyframing*-Technik erlaubt die notwendige Bewegungskontrolle der zentralen Akteure in der Animation. Daneben existieren jedoch zahlreiche Hintergrundobjekte, deren Bewegung nicht explizit spezifiziert wird und somit automatisiert erzeugt werden kann. Dies entspricht der Fragestellung der *Vorwärtsdynamik*, wie sie in dieser Arbeit untersucht wird. Die *automatisierte Bewegungsgeneration* zwischen zwei durch Keyframes vorgegebenen Objektkonfigurationen ist ein deutlich schwierigeres, weil im allgemeinen nicht eindeutiges Problem. Doch auch solche Fragestellungen der *inversen Kinematik/Dynamik* können mit Hilfe von Dynamiksimulationen beantwortet werden, indem Optimierungskriterien für die Bewegung, wie

Abbildung 1.2 Das Virtual Prototyping-Konzept.



beispielsweise die Minimierung der verrichteten Arbeit, spezifiziert werden. Neben der nur wenig aussichtsreichen, naiven Enumeration des Simulationsparameterraumes, bieten sich genetische Algorithmen zur Bestimmung einer „optimalen“ Bewegung an.

Virtual Prototyping

Die Leistungsgestaltung, also die Entwicklung neuer Produkte, gewinnt in Industriebetrieben zunehmend an Bedeutung. Die Ursache sind immer kürzer werdende Produktlebenszyklen, die mit verkürzten Zeiträumen zwischen den Markteinführungen neuer Produkte einhergehen. „*Time to Market*“ ist in vielen Branchen wie beispielsweise der Automobilindustrie ein kritischer Wettbewerbsfaktor, da der erzielbare Umsatz aufgrund niedrigerer Preisrendite und Marktsättigung im Zeitablauf sinkt. Diese vom Markt vorgegebene Anforderung erzwingt eine drastische Verkürzung der Entwicklungszeit neuer Produkte, was mit der notwendigen Änderung der organisatorischen Abwicklung des Produktentwicklungsprozesses einhergeht.

Neben der Parallelisierung der Produktentwicklung (*Concurrent Engineering*) wurde in diesem Zusammenhang das Konzept des *Virtual Prototyping* geboren. Dabei werden anstatt eines realen, zunächst ein oder mehrere virtuelle Prototypen über die Phasen des Produktentwicklungsprozesses hinweg modifiziert und evaluiert. Der virtuelle Prototyp ist bereits zur Konstruktion in Form der CAD-Daten vorhanden und wird daher mit jeder Designmodifikation entwicklungsbegleitend aktualisiert. Die Vorteile dieser Vorgehensweise liegen auf der Hand. Zum einen können bisher sequentiell ablaufende Phasen des Produktentwicklungs- und Fertigungsvorbereitungsprozesses in die frühe Konstruktionsphase vorverlagert werden. Zweitens erlaubt der Einsatz virtueller Prototypen ganz nach dem Prinzip der *präemptiven Qualitätssicherung*, „Fehlervermeidung hat Vorrang vor Fehlerentdeckung und -beseitigung“, fundamentale

Abbildung 1.3 Das Wechseln einer Glühbirne (links) und der Einbau eines Autoradios (rechts) bei einem Mercedes in einer Virtual Reality Umgebung.

Grundsätze des *Total Quality Managements* im Unternehmen umzusetzen. Beim Virtual Prototyping-Konzept steht die frühzeitige Optimierung des Produktes durch einen *Verbesserungskreislauf* aus Entwurf bzw. Modifikation des CAD-Modells und Evaluation dieses virtuellen Prototypen hinsichtlich der Einhaltung vordefinierter Produkteigenschaften im Mittelpunkt (Abbildung 1.2). Die Produkteigenschaften werden dabei durch das Marketing in Form von Produktmerkmalen vorgegeben oder stellen fertigungsbedingte Anforderungen wie Montierbarkeit oder Belastungsgrenzen der MontagearbeiterInnen dar. Gleichzeitig können kostenminimal verschiedene Produktalternativen konkurrierend entwickelt werden, wobei über die Prototypenevaluation die letztliche Produktentscheidung zu treffen ist. Die Analyse des virtuellen Prototypen erlaubt zudem sehr kostenwirksame Entscheidungen wie beispielsweise die Materialauswahl zu treffen, ohne eine Vielzahl realer Prototypen erstellen zu müssen.

Der Erfolg dieses Konzeptes hängt somit sehr stark von der Aussagekraft des Evaluationswerkzeuges ab. Kosten- und zeitintensive Experimente mit realen Prototypen können nur dann vermieden werden, wenn die Simulationsdaten auf Basis der CAD-Modelle verläßlich sind. Im Rahmen der Funktionalitätsanalyse und -bewertung spielen Dynamikdaten eine zentrale Rolle. Die Untersuchung von Reibungseigenschaften oder die Wirkung von Krafteinflüssen erlauben es, nichtanforderungsgerechte Produkteigenschaften oder gar Fehlfunktionen zu identifizieren. Des weiteren können durch virtuelle Experimente über dem Parameterraum erlaubte Toleranzen festgelegt werden, die sich in der Materialauswahl sowie in der Gestaltung der Fertigungsprozesse auswirken.

Dynamiksimulationen können unabhängig davon zur Bewertung der bereits während der Konstruktion geplanten Montageabläufe eingesetzt werden (*Virtual Assembly Planning*). Denn erst eine Berücksichtigung von Krafteinwirkungen auf die Bewegung der Montageobjekte erlaubt verläßliche Informationen über die Montierbarkeit

und die dabei auftretenden Belastungen der MontagearbeiterInnen (*Ergonomiestudien*). Man denke beispielsweise an die in Abbildung 1.3 dargestellten Ein- bzw. Ausbauoperationen, die im Automobilbereich evaluiert werden müssen. Ohne die Simulation der Wirkung auftretender Kontaktkräfte auf die Bewegung der Glühbirne bzw. des Autoradios und ohne die Berücksichtigung der eingeschränkten Bewegungsmöglichkeiten des Arms bzw. der Hand, ist eine Bewertung der Arbeitsvorgänge hinsichtlich der Schwierigkeit ihrer Ausführung undenkbar.

Die vom Simulationssystem ermittelten Daten müssen in Virtual Prototyping-Systemen zunächst aufbereitet werden und können anschließend als Modifikationsnotwendigkeiten an das CAD-System zurückgegeben werden. Erst wenn der virtuelle Prototyp einen zufriedenstellenden Entwicklungsgrad erreicht hat, wird ein realer Prototyp erstellt, der bei erfolgreicher Umsetzung dieses Verbesserungskreislaufes den vorgegebenen Qualitäts- und Funktionsansprüchen genügt.

1.3 Gliederung der Arbeit

Teil I: Einführung in die Dynamiksimulation starrer Körper

Die Dynamiksimulation starrer Körper ist ein interdisziplinäres Gebiet. Sie vereint Theorien der klassischen Mechanik, Konzepte der algorithmischen Geometrie und Robotik sowie Ergebnisse aus dem Bereich der Computergraphik. Daher haben wir im Rahmen von **Teil I** großen Wert auf die Darstellung der Grundlagen und bisher erzielten Ergebnissen in den einzelnen Problembereichen gelegt. In **Kapitel 2** wollen wir zunächst das Verständnis der Aufgabenstellung und insbesondere der in ihr formulierten konzeptionellen Spezialisierungen vertiefen. Diese betreffen das der Betrachtung zugrundeliegende *physikalische Körpermodell* sowie die Schlüsselkomponenten der Dynamiksimulation, nämlich die *Kollisionsauflösung* und *-erkennung*.

Der Begriff *impulsbasierte Dynamiksimulation* stellt auf einen speziellen Ansatz zur Berechnung der Kollisionsreaktion ab, der von MIRTICH [Mir96b] entwickelt wurde und sich aus zweierlei Gründen für eine Untersuchung im Rahmen dieser Diplomarbeit angeboten hat. Zum einen ergänzt er auf exzellente Weise einen alternativen Ansatz auf der Basis von Zwangsbedingungen, zu deren Entwicklung entscheidende Beiträge am Lehrstuhl von Professor Hotz geleistet wurden. Die Stärken und Schwächen der beiden Verfahren stellen sich vollkommen komplementär dar, so daß die Verfügbarkeit einer Implementierung beider Konzepte im Rahmen der Simulationsbibliothek SILVIA vielversprechend erschien. Daneben war die bisherige Implementierung von MIRTICH durch das eingesetzte Kollisionserkennungsystem hinsichtlich ihrer praktischen Verwendbarkeit stark eingeschränkt. Das von ihm gewählte Verfahren setzt eine Zerlegung der Simulationsobjekte in konvexe Polyeder voraus, ein Prozeß, der im allgemeinen interaktiv durchgeführt werden muß, da eine Automatisierung aufgrund der Komplexität entsprechender Algorithmen nicht sinnvoll erscheint.

Im Abschnitt 2.2 wollen wir den *impuls-* und *zwangsbasierten Ansatz* hinsichtlich der zugrundeliegenden Modelle sowie avisierten Ziele vorstellen und bezüglich ihrer Stär-

ken und Schwächen vergleichen. Wir stellen die konzeptionelle Beschreibung des Kollisionsauflösungsansatzes den eigentlichen Grundlagen voran, da die Entscheidung für einen speziellen Ansatz großen Einfluß auf die Ausgestaltung des Gesamtsystems hat. Das gewählte Verfahren bestimmt beispielsweise das verwendete Körpermodell und übt zudem großen Einfluß auf die Organisation des Simulationsablaufs aus.

Die zweite Spezialisierung, die wir in unserer Aufgabenstellung vornehmen, ist die Beschränkung auf *starre*, d.h. *nichtdeformierbare Körper*. Diese Einschränkung ist ein unumgänglicher Tribut an die Echtzeitanforderungen des Gesamtsystems, da die physikalisch korrekte Berechnung des Kontaktverhaltens sowie der daraus resultierenden Deformationen im Fall verformbarer Körper ungleich schwieriger ist. Aus dem physikalischen Modell des starren Körpers leiten wir ein *mathematisches und geometrisches Modell* ab, indem wir weitere Eigenschaften realer Körper, die sich aus unserem intuitiven Verständnis ergeben, in die Definition einbringen. Als geometrische Repräsentation wählen wir eine Polyederdarstellung, da Polyeder reale Körper beliebig genau approximieren können und ihre begrenzenden Oberflächenelemente algorithmisch einfach zu handhaben sind.

Wir verbleiben zunächst auf der rein geometrischen Betrachtungsebene und beschäftigen uns mit dem Begriff der *Bewegung* eines starren Körpers. In diesem Zusammenhang identifizieren wir ausgehend von unseren intuitiven Vorstellungen eine bestimmte Klasse von *affinen Abbildungen* als erlaubte Bewegungstransformationen. Diese Gruppe der sogenannten *eigentlichen Bewegungen* läßt sich, wie wir sehen werden, durch die Hintereinanderausführung zweier ausgezeichneter Vertreter, nämlich Translation und Rotation, erzeugen.

Die physikalische Betrachtungsweise bestätigt diese Beobachtung und führt die notwendigen Größen zur Beschreibung der *Kinematik* starrer Körper ein. Da wir die Bewegung der Objekte *simulieren* wollen, dürfen wir nicht nur nach den Beschreibungsmöglichkeiten von Bewegungen fragen, sondern müssen die Ursachen und deren Wirkungen auf die kinematischen Eigenschaften des starren Körpers untersuchen. Aus diesem Grund leiten wir die *allgemeinen Bewegungsgleichungen* her, die die Bewegung eines starren Körpers unter dem Einfluß externer Kräfte beschreiben. Der impulsbasierte Ansatz sieht jedoch als einzige externe und nicht kontaktinduzierte Kraft die Gravitation vor (*ballistische Bewegungsbahn*), so daß dieses fundamentale Differentialgleichungssystem vereinfacht werden kann.

Die zweite Schlüsselkomponente der Dynamiksimulation ist die *Kollisionserkennung*. Der Begriff „Kollisionserkennung“ wird in der Literatur sehr weit gefaßt und läßt sich letztlich auf eine *statische* bzw. *dynamische Problemstellung* zurückführen. Wir werden zunächst die wesentlichen theoretisch relevanten Ergebnisse auf diesem Gebiet vorstellen, die jedoch keinen Einzug in konkrete Implementierungen gefunden haben. Aufgrund der großen Bedeutung der Kollisionserkennung für eine Vielzahl von Anwendungsgebieten existieren zahlreiche, in der Praxis bewährte Ansätze, so daß es sich anbietet, die bisherigen Arbeiten zu klassifizieren. Im Umfeld der Dynamiksimulation ergeben sich schließlich besondere Anforderungen an die Kollisionserkennung, die wir zum Abschluß des ersten Kapitels aufzeigen wollen.

Teil II: Ein dynamischer Kollisionserkennungsansatz auf der Basis statischer Abstandsberechnung

Nachdem wir in Teil I mit den Bewegungsgleichungen ein Differentialgleichungssystem kennengelernt haben, dessen Integration die Berechnung der Bewegung starrer Körper auf ballistischen Bahnen erlaubt, verbleiben nur noch Kollisionen zwischen den Körpern als unberücksichtigte Einflußfaktoren auf das kinematische Verhalten der simulierten Objekte. Kollisionen müssen also erkannt und aufgelöst werden.

Teil II der Arbeit behandelt die Entwicklung eines *dynamischen Kollisionserkennungsverfahrens*. Das Verfahren muß dynamischer Natur sein, da wir Körperbewegungen analysieren und die Kollisionserkennung nur dann verläßlich ist, wenn sie nicht zu festen Zeitpunkten entscheidet, ob sich Simulationsobjekte überlappen, sondern ganze Zeitintervalle auf Kollisionen hin überprüft. Das von uns verwendete Konzept basiert auf der Berechnung *unterer Kollisionszeitschranken*. Ausgehend von den aktuellen kinematischen Eigenschaften bestimmen wir den frühsten Zeitpunkt, zu dem eine Überlappung der Simulationsobjekte im Rahmen ihrer Bewegung möglich ist. Die Bestimmung dieser Schranken erfordert Informationen über den Abstand der Körper bzw. eine konservative Abschätzung von diesem.

Aus diesem Grund beschäftigen wir uns in **Kapitel 3** mit der *statischen Abstandsberechnung* starrer Körper. Wir definieren zunächst den Distanzbegriff, wobei wir der Definition das Euklidische Abstandsmaß zugrunde legen. Neben der Distanzinformation sollte die Abstandsberechnung ein Punktepaar als Zeugen des Abstandsminimums bestimmen, so daß wir im Fall einer Kollision einen Kontaktpunkt ermitteln können. Anschließend wollen wir einen Überblick über die in der Literatur vorgestellten Abstandsberechnungsverfahren für polyederförmige Körper geben. *Konvexität* ist dabei eine sehr vorteilhafte, geometrische Eigenschaft, was die empirische Laufzeit der feature- und simplexbasierten Verfahren beweist. Ansätze, die diese Algorithmen zur Behandlung des allgemeinen Falls erweitern, sind für reale Anwendungen kaum praktikabel. Wesentlich vielversprechender erscheinen dagegen die Algorithmen, die auf der Basis konservativer Objektapproximationen, den sogenannten *Hüllkörpern*, arbeiten.

Wir formulieren das statische Abstandsberechnungsproblem zunächst als kombinatorisches Optimierungsproblem und stellen das generische *Branch-and-Bound-Verfahren* zur Lösung solcher Fragestellungen vor. Zur Umsetzung dieses Verfahrens benötigen wir verschiedene Operationen und Datenstrukturen, die wir in den sich anschließenden Abschnitten erläutern werden.

Die *elementaren Abstandsberechnungen* erlauben die Bestimmung der Distanz zweier Oberflächenelemente der beiden Körper und liefern somit eine *obere Schranke* für den Abstand des betrachteten Körperpaares. Anschließend werden wir uns mit dem Hüllkörperkonzept vertraut machen, das eine entscheidende Rolle in der Lösung des Abstandsberechnungsproblems übernimmt.

Wir werden die Bedeutung der inneren bzw. äußeren Körperapproximation als notwendiges bzw. hinreichendes Kriterium der Kollisionsfreiheit zweier Körper vorstellen und den Einsatz hierarchisch verfeinerter Hüllkörperüberdeckungen, sogenann-

ter *Hüllkörperhierarchien*, als Lieferant unterer Abstandsschranken erläutern. Als Hüllkörpertypen kommen nur sehr einfache geometrische Objekte in Frage, die man im allgemeinen als *Primitive* bezeichnet. Diese sollten die einzuschließende Flächenmenge möglichst genau approximieren, um präzise Abstandsschranken liefern zu können. Aus diesem Grund werden wir uns mit der Messung der *Approximationsgüte* von Hüllkörpern beschäftigen und dabei das Volumen als Kompromiß aus einfacher Berechenbarkeit und Aussagekraft der Hüllkörperkonstruktion zugrunde gelegt.

Beim Aufbau der Hüllkörperhierarchie wird die Flächenmenge des Körpers immer feiner partitioniert und jede Unterteilung mit neuen Hüllkörpern überdeckt. Die präziseste Überdeckung des Körperrandes erhält man schließlich durch die Verpackung einzelner Flächen. In diesem Zusammenhang werden wir verschiedene Vorgehensweisen zum Aufbau der Hierarchie vorstellen, wobei der *Aufteilungsstrategie* der Flächenmenge eine besondere Bedeutung zukommt. Dabei muß nämlich die geometrische Anschauung der Flächenmengenpartitionierung, die Zerlegung des Objektes in Teilkörper, berücksichtigt werden.

Eine zentrale Operation des Branch-and-Bound-Verfahrens ist die Berechnung *unterer Schranken* für die Distanz zweier Flächenmengen. Eine solche Schranke erhält man in Form des Abstands der Hüllkörper, welche die Flächenmengen einschließen. Da wir das Abstandsberechnungsverfahren zu diskreten Zeitpunkten der Körperbewegung aufrufen, muß die Hüllkörpergeometrie vor der Distanzberechnung an die Bewegung des Körpers, d.h. der eingeschlossenen Flächenmenge, angepaßt werden. Diese *Aktualisierungoperation* kann relativ teuer sein, wenn das Primitiv in der Eigenschaft als volumenminimaler Hüllkörper unter den Bewegungsabbildungen nicht abgeschlossen ist.

Die Effizienz des Branch-and-Bound-Verfahrens hängt neben der Verfügbarkeit präziser unterer Schranken sehr stark von der *Durchmusterungsstrategie* des Entscheidungsbaumes ab. In diesem Zusammenhang haben sich *Greedy-Strategien* als besonders effizient herausgestellt. Das von uns entwickelte globale Greedy-Verfahren eignet sich in ganz hervorragender Weise zur verläßlichen Erfüllung von Echtzeitanforderungen. Es ist in der Lage, eine untere Schranke für den Abstand der beiden Körper zu jedem Zeitpunkt der Entscheidungsbaumtraversierung aufrechtzuhalten. Ist ein vorgegebenes Zeitbudget ausgeschöpft, oder weichen untere und obere Abstandsschranke nur noch geringfügig voneinander ab, so kann das Branch-and-Bound-Verfahren vorzeitig abgebrochen und anhand der konservativen Abstandsinformation valide Kollisionszeitschranken bestimmt werden.

Im Rahmen der Beschleunigungsansätze wollen wir zunächst das *Repräsentantenkonzept* vorstellen, das eine zusätzliche Aktualisierung der oberen Schranken in den inneren Knoten des Entscheidungsbaumes ermöglicht. Um der *temporären Kohärenz* der Problemstellung im dynamischen Umfeld Rechnung zu tragen, speichern wir die nahesten Punkte des letzten Zeitschritts. Der Abstand dieser beiden Punkte stellt beim nächsten Aufruf des Verfahrens einen guten Startwert für die obere Schranke dar. Begnügt man sich mit *approximativen Abstandsinformationen*, so läßt sich eine weitere, deutliche Beschleunigung der Abstandsberechnung erzielen. Zur Bestimmung des frühsten Kollisionszeitpunktes zweier Körper machen selbstverständlich nur untere Abstands-

schranken Sinn. Aus diesem Grund werden ausschließlich Verfahren zur Berechnung solcher konservativer Abstandsinformationen vorgestellt. Diese ergeben sich durch Modifikationen der bereits erwähnten Traversierungsstrategien.

In **Kapitel 3** wird das statische Abstandsberechnungsverfahren zu einer *dynamischen Kollisionserkennung* ausgebaut. Wir beschreiben zunächst die Problemstellung als Nullstellensuche auf der zeitparametrisierten *Distanzfunktion*. Mit Hilfe unterer *Kollisionszeitschranken* kann der frühste Kollisionszeitpunkt ausgehend von einer überlappungsfreien Konfiguration der Objekte *konservativ* und mit *beliebiger Genauigkeit* approximiert werden.

Während diese Form der Nullstellensuche für den Zweikörperfall algorithmisch sehr einfach beschrieben werden kann, benötigen wir für den n-Körperfall eine spezielle Datenstruktur, die wir als *Kollisionsheap* bezeichnen. Dabei handelt es sich um eine Prioritätswarteschlange, die die Objektpaare der Simulation bezüglich ihres frühsten Kollisionszeitpunktes ordnet. Um den Verwaltungsaufwand der Datenstruktur zu verringern, stellt man nur solche Paare in den Kollisionsheap ein, die sich in dem zu simulierenden Zeitschritt „nahe“ kommen.

Die entsprechenden Objektpaare können mit Hilfe von *Bewegungshüllkörpern*, d.h. konservativen Approximationen des durch ihre Bewegungen überschrittenen Volumens identifiziert werden. Wir verwenden in diesem Zusammenhang achsenorientierte Boxen und bezeichnen zwei Objektpaare als „nahe“ innerhalb eines Zeitintervalls, wenn sich ihre Bewegungshüllkörper schneiden. Zur effizienten Bestimmung aller überlappenden Hüllkörper bietet sich ihre Projektion auf die Koordinatenachsen bzw. -ebenen an. Als Überlappungstest ist dann lediglich ein ein- bzw. zweidimensionaler *Sweepalgorithmus* erforderlich. Ein ganz anderes Verfahren ist das von MIRTICH erweiterte *Raumpartitionierungsverfahren* von OVERMARS. Die Lokalisation der Bewegungshüllkörper mit Hilfe hierarchisch organisierter Hashtabellen erlaubt potentiell überlappende Hüllkörperpaare effizient zu bestimmen. Dieses Verfahren kann durch Ausnutzung temporärer Kohärenz in der Praxis weiter beschleunigt werden.

Abschließend werden wir den Ablauf der *Simulationsschleife* beschreiben, die durch die Kollisionserkennung gesteuert wird.

Teil III: Die Kollisionsauflösung

Zur Vervollständigung des Dynamiksimulationssystems benötigen wir neben der Kollisionserkennung eine zweite Schlüsselkomponente, deren Aufgabe die Berechnung der *Kollisionsreaktion* ist. In **Kapitel 5** stellen wir das von uns implementierte Verfahren von MIRTICH vor, das zweifellos als das theoretisch fundierteste, impulsbasierte Simulationskonzept gilt. Wir werden zunächst das zugrundeliegende Kontaktmodell anhand seiner zentralen Annahmen vorstellen. Die Unterstellung einer *infinitesimal kleinen Kollisionszeit* ist die physikalische Rechtfertigung für die Berechnung von Impulsen statt Kontaktkräften als Basis der Kollisionsreaktion. Das impulsbasierte Modell berücksichtigt Reibung zwischen den Körpern durch Integration des *Coulombschen*

Reibungsgesetzes in die dynamische Kollisionsanalyse. Mikroskopische Deformationen werden durch einen einfachen Koeffizienten berücksichtigt, der den Bereich zwischen vollkommener *Elastizität* und *Plastizität* der Körper abdeckt. Die Energieumwandlung, die zwischen der *Kompressions-* und *Rückstellungsphase* einer Kollision stattfindet, wird mit Hilfe der *Strongeschen Hypothese* beschrieben.

Abschnitt 5.2 führt ein Koordinatensystem ein, das eine vereinfachte Kollisionsanalyse ermöglicht. Des weiteren wird die Frage beantwortet, wie physikalische Kollisionen von bereits als Kontakt aufgelösten, geometrischen Durchdringungen unterschieden werden können.

Anschließend analysieren wir die Kollisionsdynamik und leiten mit den *Kollisionsgleichungen* eine Beziehung zwischen Impulsänderung und Änderung der Relativgeschwindigkeit der beiden Objekte im Kontaktpunkt her. Letztere kann durch ein Differentialgleichungssystem beschrieben werden, das wir im Rahmen der *Kollisionsintegration* lösen. Da die Zeit aufgrund der modelltheoretischen Annahme über die Kollisionsdauer als Integrationsparameter ungeeignet ist, müssen andere physikalische Größen hinsichtlich ihrer Eignung zur *Kollisionsparametrisierung* evaluiert werden.

Die Energieveränderung des dynamischen Systems durch die Kollisionsauflösung ist ein wichtiger Indikator für die Plausibilität des Kollisionsmodells. In diesem Zusammenhang konnten wir zeigen, daß die Energie des dynamischen Systems durch die impulsbasierte Kollisionsauflösung nicht zunimmt (*Energiekonsistenz*). Diese Invariante ist nicht bei allen Kollisionsauflösungsverfahren garantiert.

Abschließend behandeln wir permanente Kontaktsituationen wie Gleiten, Rollen oder Ruhen. Diese werden im impulsbasierten Modell auf spezielle Weise, nämlich als eine hochfrequente Folge sogenannter *Mikrokollisionen*, behandelt. Aus Effizienzgründen unterscheidet sich bei diesen Kontaktsituationen die Berechnung der Kollisionsreaktion von dem zuvor vorgestellten, allgemeinen Fall, wobei die Energiekonsistenzgarantie verloren geht.

Teil IV: Evaluation des Simulationsverfahrens und Integration in die Softwarebibliothek SiLVIA

In **Teil IV** beschreiben wir zunächst im Rahmen von **Kapitel 6** die Ziele und die Struktur der Softwarebibliothek SiLVIA, die am Lehrstuhl von Professor Hotz entwickelt wurde. Sie stellt neben zahlreichen Basisalgorithmen und -datenstrukturen effiziente Kollisionserkennungsverfahren und schwerpunktmäßig Algorithmen zur Echtzeitsimulation der Dynamik kollidierender, starrer Körper bereit. Im Laufe des SiLVIA-Projektes wurden die hier vorgestellten Algorithmen in die Bibliothek integriert und haben dort teilweise weitergehende Aufgaben übernommen.

Der nächste Abschnitt stellt einige Beispielsimulationen vor, die wir zum Testen der Algorithmen, zur Evaluation des Reibungsmodells und zum Vergleich der beiden zentralen Kollisionsauflösungsansätze durchgeführt haben. Es handelt sich dabei um sogenannte „mechanische Spielzeuge“, die ein charakteristisches und teilweise verblüffendes Verhalten in Gegenwart von Reibung zeigen. Da die in der Realität zu beobachtenden Effekte nur im Fall einer präzisen Reibungsmodellierung simulationstechnisch

nachzuvollziehen sind, zeigen die erfolgreichen Simulationsergebnisse die qualitative Aussagekraft der impulsbasierten Dynamiksimulation. Als erstaunlich erweist sich der quantitative Vergleich der Simulationsdaten unseres impulsbasierten Systems mit den Daten, die das zwangsbasierte Verfahren von SAUER [SS98a, SS98b] liefert. Beide Ansätze zeigen im Fall der Simulation des Steh-auf-Kreisels nahezu identische Kurven der zentralen physikalischen Größen.

In **Kapitel 7** diskutieren wir die notwendigen und denkbaren Erweiterungen des Simulationssystems in Hinblick auf mögliche Effizienzsteigerungen sowie eine bessere Integrierbarkeit in die Schlüsselanwendungen wie beispielsweise Virtual Prototyping-Systeme.

1.4 Resultate

Das zentrale Ergebnis dieser Diplomarbeit ist die Realisierung eines impulsbasierten Echtzeitsimulationssystems für die Dynamik starrer Körper. Diese Aufgabe umfaßte im wesentlichen die Eigenentwicklung einer echtzeitfähigen dynamischen Kollisionserkennungskomponente auf der Basis unterer Kollisionszeitschranken sowie die Implementierung und Evaluation des impulsbasierten Kollisionsauflösungsverfahrens von MIRTICH [Mir96b].

Zur Lösung des Kollisionserkennungsproblems haben wir einen auf der Branch-and-Bound-Strategie beruhenden Algorithmus zur Berechnung der Euklidischen Distanz zweier Körper mit polygonaler Randbeschreibung entwickelt. Das Abstandsberechnungverfahren verwendet Hüllkörperhierarchien, um auf effiziente Weise die Menge aller Flächenpaare der beiden Körper aufzuzählen. Der Algorithmus ist daher in der Lage, selbst mit schlecht konditionierten Eingabedaten wie beispielsweise einer Randbeschreibung ohne Flächeninzidenzinformationen zu operieren.

Wir haben uns nicht wie bisherige Ansätze auf einen bestimmten Hüllkörpertyp festgelegt, sondern ermöglichen den Einsatz all der Hüllkörpertypen, die sich im Rahmen der zahlreichen Ray-Tracing- und Kollisionserkennungsverfahren als effizient erwiesen haben.

Diese Flexibilität erforderte für jeden Hüllkörpertyp die Entwicklung effizienter Algorithmen zur Abstandsberechnung sowie Geometrieaktualisierung konkreter Hüllkörperpaare. Daneben waren wir bemüht, die elementaren Abstandsberechnungen zwischen den Oberflächenelementen und insbesondere zwischen zwei Flächen so effizient wie möglich zu gestalten.

Ein zentrales Ergebnis ist die Entwicklung einer Durchmusterungsstrategie der Hüllkörperhierarchien, die mit Hilfe einer geeigneten Datenstruktur eine untere Schranke für den Abstand der beiden Körper während der Traversierung aufrechterhält und verbessert. Eine solche konservative Abstandsinformation ist gerade im Zusammenhang mit der Erfüllung von Echtzeitanforderungen sehr nützlich, da sie bei Abbruch der Berechnung aufgrund eines abgelaufenen Zeitbudgets die Bestimmung gültiger

Kollisionszeitschranken erlaubt. Zusätzlich ist es möglich, durch Vergleich mit der aktuellen oberen Schranke des Branch-and-Bound-Verfahrens gütebasierte Abbruchskriterien zu formulieren.

Im Rahmen der Beschleunigungsansätze haben wir unter anderem gezeigt, wie man temporäre Kohärenz im dynamischen Umfeld ausnutzen kann, um einen guten Startwert für die obere Abstandsschranke des Branch-and-Bound-Verfahrens zu bestimmen.

Zur Berechnung des frühsten Kollisionszeitpunktes war es erforderlich, die unteren Kollisionszeitschranken, die MIRTICH für den Spezialfall konvexer Polyeder formuliert hat, auf allgemeine Körperdarstellungen zu erweitern.

Im Rahmen der Evaluation der impulsbasierten Kollisionsauflösung konnten wir einige schwierige Experimente der klassischen Mechanik durch Simulation reproduzieren. Neben diesen qualitativen Ergebnissen ließ sich eine erstaunliche quantitative Übereinstimmung mit den Simulationsdaten des Verfahrens von SAUER feststellen, welches auf dem konzeptionell unterschiedlichen, zwangsbasierten Kollisionsmodell beruht.

Danksagungen

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Günter Hotz für die Vergabe des interessanten Themas und die hervorragenden Arbeitsbedingungen am Lehrstuhl für Angewandte Mathematik und Informatik bedanken. Dank gebührt ihm nicht nur für die Möglichkeiten, die er mir an seinem Lehrstuhl geboten hat, sondern auch für die Ratschläge und Anregungen, die ich während meiner Arbeit erfahren habe.

Insbesondere danke ich Herrn Dr. Elmar Schömer, der mich in das Themengebiet dieser Arbeit eingeführt und bis zur Abgabe der Diplomarbeit ausgezeichnet betreut hat. Die unzähligen Gespräche mit ihm haben mir stets hilfreiche Impulse vermittelt und mein Verständnis der zu lösenden Probleme erweitert. Er hat somit wesentlich zum Gelingen der Arbeit beigetragen.

Herrn Dr. Frank Follert, Herrn Dr. Thomas Chadzelek und besonders Herrn Jörg Sauer danke ich für die hervorragende Betreuung während der Seminare bzw. dem Fortgeschrittenenpraktikum. Sie standen jederzeit für Fragen und Diskussionen zur Verfügung und ihre wertvollen Ratschläge haben mich bei der Erstellung dieser Arbeit unterstützt.

Dank sagen möchte ich auch allen Mitarbeitern und Diplomanden am Lehrstuhl von Prof. Hotz, insbesondere dem SILVIA-Team, namentlich Andres Kerzmann, Bastian Kleineidam, Rainer Schmid und Thomas Warken. Die kollegiale und freundschaft-

liche Atmosphäre in unserer Gruppe habe ich stets sehr geschätzt. Die Diskussionen und gemeinsam gemachten Erfahrungen haben die vorliegende Arbeit maßgeblich beeinflußt.

Herzlichst danken möchte ich auch meinem Kommilitonen, Sven Thiel, mit dem ich seit dem ersten Tag meines Studiums befreundet bin. Unsere Zusammenarbeit während des Studiums fand sicherlich Niederschlag in dieser Arbeit.

Bei Herrn Prof. Dr. Kurt Mehlhorn möchte ich mich bedanken, daß er es mir ermöglichte im LEDA-Projekt mitzuarbeiten. Die im Rahmen meiner Tätigkeit gemachten Erfahrungen waren insbesondere für den praktischen Teil der Diplomarbeit sehr hilfreich.

Ein besonderer Dank gilt Raphaela Adam und Nina Wagner für die Bereitschaft und vor allem die Gewissenhaftigkeit, mit der sie diese Arbeit Korrektur gelesen haben. Bedanken möchte ich mich aber auch bei allen anderen Freunden und Freundinnen, die stets für mich da waren und großes Interesse an meiner Arbeit gezeigt haben.

Mein größter Dank gilt jedoch meiner Familie, auf deren Unterstützung und Verständnis stets Verlaß war. Ohne diesen Rückhalt und die Freiräume, die sie mir gewährt haben, hätte ich mich nicht derart sorgenfrei meinem Studium widmen können. Meinem Bruder Carsten wünsche ich ein ähnlich abwechslungsreiches und erfolgreiches Studium.

Einführung in die Dynamiksimulation starrer Körper

2 Grundlagen

In diesem Kapitel wollen wir dem Leser einen Einstieg in die zentralen Problemstellungen der Dynamiksimulation vermitteln. Neben einer einführenden und vergleichenden Darstellung des impuls- und zwangsbasierten Ansatzes zur Kollisionsreaktionsberechnung steht das der Arbeit zugrundeliegende Körper- und Bewegungsmodell im Mittelpunkt unserer Betrachtungen. Die Frage nach der Simulation von Bewegungen führt uns zu den Theorien der klassischen Mechanik, insbesondere in das Gebiet der Kinematik und Dynamik starrer Körper, deren zentrale Begriffe und Ergebnisse wir vorstellen werden. Ein Überblick über die theoretisch und praktisch relevanten Kollisionserkennungsansätze und eine Analyse der speziellen Anforderungen im Umfeld der Dynamiksimulation schließt die Grundlagenbetrachtung ab.

2.1 Notation

Wir werden zunächst einige Vereinbarungen hinsichtlich der in dieser Arbeit gebräuchliche Notation treffen. Dabei möchten wir aus Gründen der Übersichtlichkeit nur die Schreibweisen erläutern, die wir über alle Kapitel hinweg verwenden, während problemspezifische Bezeichnungen an entsprechender Stelle vereinbart werden.

- *Skalare* werden kursiv dargestellt. So sind beispielsweise i, n oder φ Skalare.
- *Vektoren* und *vektorwertige Funktionen* schreiben wir fett. Beispiele hierfür sind $\mathbf{a}$, $\mathbf{F}$, $\boldsymbol{\omega}$ oder $\boldsymbol{\xi}$. Dabei fassen wir einen Vektor $\mathbf{a} \in \mathbb{R}^n$ stets als Spaltenvektor auf:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} .$$

  Für den Nullvektor schreiben wir $\mathbf{0}$.
- *Matrizen* werden ebenfalls fett dargestellt. Beispiele hierfür sind $\mathbf{R}$, $\mathbf{D}$ oder $\mathbf{I}$. Dabei bezeichnen wir mit $\mathbf{E}$ die Einheitsmatrix und schreiben $\mathbf{0}$ für die Nullmatrix. Sind $\mathbf{a}_1, \ldots, \mathbf{a}_n$ Vektoren des $\mathbb{R}^3$, so stellt $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$ die Matrix mit Spaltenvektoren $\mathbf{a}_i$, $1 \leq i \leq n$ dar. Daneben werden wir häufig das Kreuzprodukt zweier Vektoren des $\mathbb{R}^3$ in Matrixschreibweise formulieren. Seien $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, dann gilt:

$$\mathbf{a} \times \mathbf{b} = \mathbf{a}^{\times}\mathbf{b} ,$$

wobei $\mathbf{a}^{\times}$ die folgende 3×3-Matrix bezeichnet:

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

Die Kreuzproduktmatrix ist schiefsymmetrisch:

$$\left(\mathbf{a}^{\times}\right)^{\mathrm{T}} = -\mathbf{a}^{\times} .$$

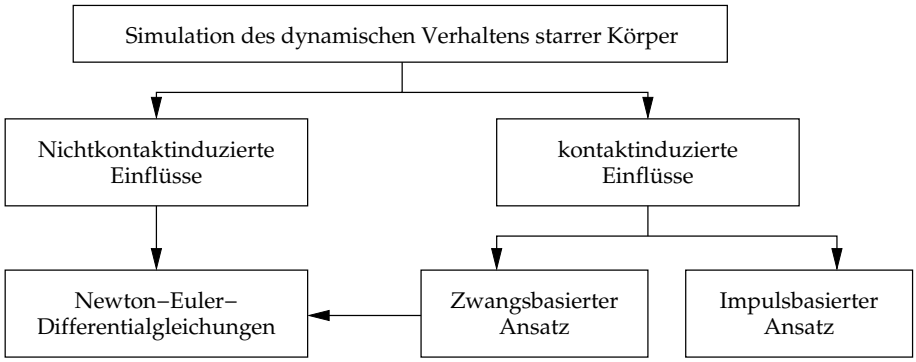
- *Punkte* unterscheiden sich in der Notation von ihren *Ortsvektoren*, da sie kursiv und nicht fett dargestellt werden.

2.2 Einführung in die Berechnung der Kollisionsdynamik starrer Körper

Die Bewegung eines physikalischen Objektes ist das Ergebnis externer Einflußfaktoren. Diese Einflußfaktoren können unterschiedlichster Natur sein. So wirken sich beispielsweise magnetische und elektrische Felder, Luftwiderstand, aber auch physikalische Kontakte mit anderen Objekten auf das kinematische Verhalten aus. Die klassische Mechanik ist dank ihres heutigen Wissensstandes in der Lage, viele Systeme physikalischer Objekte unter Berücksichtigung verschiedenster Einflußfaktoren zu analysieren und in bezug auf ihr zukünftiges Verhalten zu beschreiben. In der Literatur beschränkt man sich dabei nahezu immer auf spezielle Beispielsysteme, für die ein Modell aus der allgemeinen Theorie abgeleitet wird, das Voraussagen über das dynamische Systemverhalten erlaubt. So wird in [Nol96] beispielsweise der „kräftefreie symmetrische Kreisel“ untersucht. Eine allgemeine Dynamiksimulation, wie sie von uns angestrebt wird, muß dagegen ein *ganzheitliches Modell* bereitstellen, das die Vorhersage des Systemverhaltens in jeder Situation gestattet.

Die Bewegung nichtdeformierbarer Körper unter Einflüssen, die sich nicht auf Kontakte zurückführen lassen, ist gut verstanden und unterliegt nicht mehr der Forschung auf dem Gebiet der Dynamiksimulation. Das sogenannte NEWTON-EULER-*Differentialgleichungssystem* (vgl. Abschnitt 2.5) beschreibt die Bewegung eines starren Körpers unter dem Einfluß von externen Kräften. Während externe Einflüsse, die nicht kontaktinduziert sind (z.B. Gravitation), auf einfache Weise in diesem Konzept berücksichtigt werden können, stellt sich die Bestimmung von Kräften, die aus Kontaktwechselwirkungen resultieren, als anspruchsvolles Problem heraus. Obwohl eine realitätsgetreue Behandlung von Kontakten für die Simulationsgüte von entscheidender Bedeutung ist, sind viele Ansätze physikalisch zu ungenau oder basieren auf dubiosen Annahmen. Im Rahmen der Modellierung des Kontaktverhaltens wird der *Trade-Off zwischen Genauigkeit der Simulation und Laufzeitverhalten* offensichtlich. Crashtestanwendungen im Automobilbau zeigen zwar, daß Kontaktwechselwirkungen mit Hilfe von *Finite-Elemente-Methoden* (FEM) [Sha98] sehr präzise beschrieben werden können. Dabei sind jedoch selbst unter Einsatz leistungsfähigster Computersysteme keine für

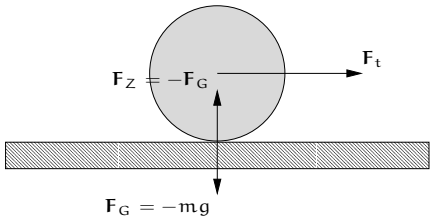
Abbildung 2.1 Die simulationstechnische Erfassung externer Einflußfaktoren auf die Bewegung starrer Körper.



Interaktion akzeptablen Simulationszeiten zu erwarten. Ist die Deformierbarkeit der Objekte allerdings unerläßlich für die zu untersuchenden Fragestellungen, so gibt es derzeit keine Alternativen zu Finite-Elemente-Methoden bzw. verwandten Ansätzen. Falls die Objektdeformationen in der Realität weder sichtbar sind noch Einfluß auf das qualitative Verhalten der Objekte haben, ist der Einsatz dieser Verfahren lediglich für solche Anwendungen angezeigt, in denen die *quantitative Genauigkeit* der Simulationsdaten im Mittelpunkt steht und Anforderungen an die Simaultionszeit hinter diesem zentralen Ziel zurückstehen müssen.

Strebt man auf der anderen Seite *Echtzeitabläufe* an, so führt kein Weg daran vorbei, die Deformierbarkeitsannahme fallen zu lassen. Diese Einschränkung ist oft akzeptabel, da in vielen real motivierten Fragestellungen Körper miteinander in Kontakt treten, ohne daß eine Änderung ihrer Form beobachtbar ist oder sich auf ihr qualitatives Verhalten auswirkt. Kritisch wird die *Starrheitsannahme* nur dann, wenn mikroskopische Verformungen das beobachtbare Verhalten der simulierten Körper beeinflussen. Qualitativ korrekte Aussagen genügen in vielen Anwendungen, so beispielsweise bei der Frage, ob ein Montagevorgang durchführbar ist oder nicht. Hier erwartet man stattdessen kurze Antwortzeiten des Simulationssystems, um den Montageprozeß oder die zu montierenden Objekte derart zu modifizieren, daß das gewünschte Ziel der Montierbarkeit erreicht wird. Viele Anwendungen erfordern also Verfahren der Kontaktbehandlung, die echtzeitfähige Simulationen erlauben und dennoch über eine hohe physikalische Genauigkeit verfügen. In diesem Zusammenhang haben sich zwei Ansätze durchgesetzt, die auf völlig unterschiedlichen Kontaktmodellen beruhen und entsprechend diesen als *zwangsbasierte* bzw. *impulsbasierte Simulationsmethode* bezeichnet werden. Der letztere Ansatz liegt dieser Arbeit zugrunde. Um die Stärken und Schwächen der impulsbasierten Simulationsmethode besser verstehen zu können, wollen wir zunächst das zwangsbasierte Kontaktmodell und die darauf beruhenden Simulationsverfahren erläutern. Abbildung 2.1 faßt diesen Überblick über das Forschungsgebiet zusammen.

Abbildung 2.2 Das physikalische Konzept der Zwangskräfte.



2.2.1 Der zwangsbasierte Ansatz

Im Rahmen dieses Ansatzes werden Kräfte bestimmt, die die Einhaltung bestimmter Zwangsbedingungen sicherstellen und daher als Zwangskräfte bezeichnet werden. Um das Konzept zu veranschaulichen, betrachten wir das folgende Beispiel, welches in Abbildung 2.2 dargestellt ist:

Beispiel 2.1. Ein Ball rollt auf einer Ebene. Auf den Ball wirkt die Gravitationskraft. Diese Kraft wird kompensiert durch eine Zwangskraft, die die Ebene auf den Ball ausübt. Sie erzwingt eine Nichtdurchdringung von Ball und Ebene, indem sie betragsmäßig der Gravitationskraft entspricht, ihr jedoch entgegengerichtet ist. □

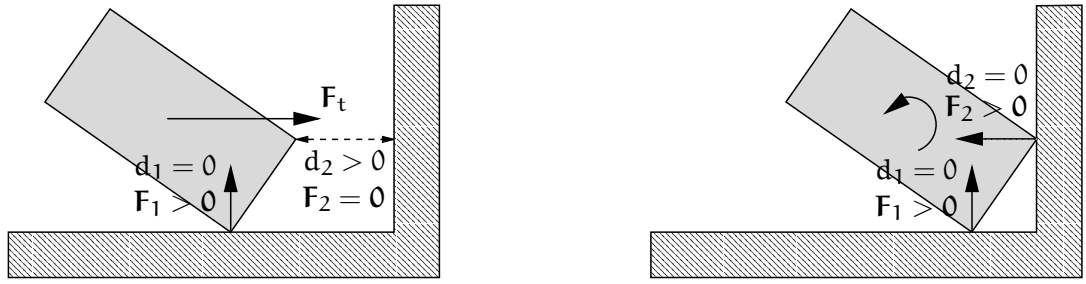
In diesem speziellen Fall ist die Zwangskraft von der eigentlichen Bewegung des Körpers unabhängig, was selbstverständlich im allgemeinen nicht gegeben ist. Das Beispiel täuscht ein wenig über die Tatsache hinweg, daß das Konzept der Zwangsbedingungen und der daraus resultierenden Kräfte genereller Natur ist und sich nicht nur auf den Kontaktfall beschränkt. Betrachtet man beispielsweise gelenkig verbundene Körper, so schränken Bedingungen, die durch die Gelenke induziert werden, die Pendelbewegung der Körper ein. Unser Interesse gilt jedoch primär der Kontaktbehandlung, so daß wir uns ausschließlich mit Kontaktbedingungen und den resultierenden Zwangskräften beschäftigen wollen. Diese Kräfte bezeichnet man daher als Kontaktkräfte. Es lassen sich im wesentlichen zwei Ansätze zur Kontaktmodellierung mittels Zwangsbedingungen unterscheiden. Dabei handelt es sich zum einen um die sogenannten *Penalty Methoden* und zum anderen um die *analytischen Methoden*.

Die Penalty Methoden

Die zentrale Idee dieses Ansatzes, der auf eine Arbeit von [MW88] zurückgeht, ist die Einführung einer am Kontaktpunkt ansetzenden Feder. Diese liefert abhängig von der Durchdringung der kollidierenden Körper eine Federkraft, die die Objekte auseinander drückt. Nach der Auflösung der Kollision wird die Feder wieder entfernt.

Das Verfahren ist nicht unproblematisch, wie die folgenden Überlegungen zeigen. Die Wahl der Federkonstanten gestaltet sich schwierig, da hier sowohl die Durchdringungstiefe der Körper, als auch der Einfluß der Körpermassen berücksichtigt werden muß. Des weiteren sind häufig große Federkonstanten erforderlich, die die numerische

Abbildung 2.3 Die Entstehung neuer Kontaktbedingungen.



Erster Kontaktpunkt. Zweiter Kontaktpunkt.

Stabilität der zugrundeliegenden Gleichungen einschränken [WGW90]. Da die Kontakte erst nach Feststellung einer positiven Durchdringungstiefe aufgelöst werden, sind die berechneten Kontaktkräfte im allgemeinen zu groß. Diesen Schwachpunkten steht gegenüber, daß das Verfahren schnell und einfach zu implementieren ist. Es findet seine Anwendung beispielsweise im Animationsbereich, wo ein *realistischer Eindruck* den gestellten Anforderungen genügt.

Die analytischen Methoden

Die analytischen Methoden gehen im wesentlichen auf Arbeiten von LÖTSTEDT [Löt81, Löt82, Löt84] und BARAFF [Bar89, Bar92, Bar94] zurück. Die von diesen Verfahren berechneten *Kontaktkräfte* nehmen über die Bewegungsgleichungen Einfluß auf die Körperbewegung im nächsten Zeitschritt. Das Beispiel aus Abbildung 2.3 verdeutlicht das Konzept.

Beispiel 2.2. Ein Quader wird entlang einer Ebene auf eine Wand hin geschoben. Zunächst ist lediglich die Kontaktbedingung bezüglich der Ebene zu beachten. Sobald der Quader jedoch an der Wand anstößt, muß ein zweiter Kontaktpunkt und somit eine neue Kontaktbedingung berücksichtigt werden. Die beiden resultierenden Kontaktkräfte verursachen eine Rotationsbewegung des Körpers, falls der Quader weiter auf die Wand zu geschoben wird. □

LÖTSTEDT und BARAFF beschreiben die Kontaktbedingungen in Form eines *Linearen Komplementaritätsproblems* (LCP), dessen Lösung die Kontaktkräfte $\mathbf{F} = (F_1, \ldots, F_k)^T$ in den k Kontaktpunkten sind. Bezeichnet $\mathbf{d}(\mathbf{F}) = \left[d_1(F_1, \ldots, F_k), \ldots, d_k(F_1, \ldots, F_k)\right]^T$ die aktuellen Kontaktdistanzen, d.h. die Abstände der beiden Körper in den k Kontaktpunkten, so hat die LCP-Formulierung von BARAFF [Bar94] die folgende Form:

$$\ddot{\mathbf{d}}(\mathbf{F}) = \mathbf{A}\mathbf{F} - \mathbf{b} \geq 0, \quad \mathbf{F} \geq 0, \quad \mathbf{F}^T \ddot{\mathbf{d}}(\mathbf{F}) = 0\,. \tag{2.1}$$

Dabei stellt der relative Beschleunigungsvektor $\ddot{\mathbf{d}}(\mathbf{F})$ eine lineare Funktion in $\mathbf{F}$ dar. A und b sind durch die Kontaktgeometrie eindeutig bestimmt. Die Bedingungen

$\ddot{\mathbf{d}}(\mathbf{F}) \geq \mathbf{0}$ und $\mathbf{F} \geq \mathbf{0}$ stellen sicher, daß die Körper sich im Kontaktpunkt nicht aufeinanderzu beschleunigen und daß die Kontaktkräfte $\mathbf{F}$ die Körper auseinander drücken. Die Komplementaritätsbedingung $\mathbf{F}^T\ddot{\mathbf{d}}(\mathbf{F}) = 0$ verhindert das Wirken einer Kontaktkraft zwischen zwei voneinander weg beschleunigten Körpern. Die zentralen Fragen in diesem Zusammenhang zielen auf die *Existenz* sowie die *Eindeutigkeit* von $\mathbf{F}$. Falls keine Reibung berücksichtigt wird und keine Kontaktdegeneration vorliegt, so hat das LCP 2.1 stets eine eindeutige Lösung [Bar92].

Ein großer Nachteil der LCP-Formulierung besteht darin, daß die Kontaktbedingung $\mathbf{d}(\mathbf{F}) \geq \mathbf{0}$, welche die Nichtdurchdringbarkeit der Körper garantiert, aufgrund der Zeitdiskretisierung nicht exakt eingehalten werden kann. Die Akkumulation dieses Fehlers erweist sich als ernsthaftes Problem. Um diesen Nachteil zu beseitigen wird in [BS98a, SS98a] die Formulierung eines *nichtlinearen Komplementaritätsproblems* (NCP) vorgeschlagen:

$$\mathbf{d}(\mathbf{F}) \geq \mathbf{0}, \quad \mathbf{F} \geq \mathbf{0}, \quad \mathbf{F}^T\mathbf{d}(\mathbf{F}) = 0\,. \tag{2.2}$$

Die Abhängigkeit der Kontaktdistanzen d von den ermittelten Kräften kann selbstverständlich nicht mehr als lineare Funktion dargestellt werden.

Zur Lösung des NCPs bieten sich zwei Vorgehensweisen an. Mit Hilfe der FISCHER-Funktion $\varphi : \mathbb{R}^2 \to \mathbb{R}$, $(a, b) \mapsto \sqrt{a^2 + b^2} - a - b$, die die Eigenschaft

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

erfüllt, kann die Lösung des Ungleichungssystems 2.2 auf die eines Gleichungsystemes reduziert werden [Kan96]: Alternativ dazu kann das NCP linearisert [ST95, SS98a] und mit Hilfe des *Lemke Algorithmus'* [Lem65, CPS92] oder *Feasible-Interior-Point-Methoden* gelöst werden [TPL96]. Benutzt man ein *Fixpunkt-Iterationsschema*, so konvergiert die Folge von LCPs im allgemeinen schnell gegen die gesuchte Lösung des NCPs.

Das Coulomb-Gesetz (vgl. Abschnitt 5.1.3) erlaubt die Berücksichtigung von Gleit- und Haftreibung in der LCP- bzw. NCP-Formulierung. Dazu wird der Reibungskegel, der im Fall von Haftreibung zu betrachten ist, linear approximiert. Diese Facettierung ist im Rahmen der LCP- bzw. der linearisierten NCP-Formulierung unumgänglich [SS98a]. Im Reibungsfall ist weder die Existenz noch die Eindeutigkeit von $\mathbf{F}$ garantiert.

Als potentielles Problem dieser Verfahren gilt die Abhängigkeit von temporärer Kohärenz (vgl. Abschnitt 2.6.4). Mangelnde Eindeutigkeit der Kontaktkräfte im Fall degenerierter Kontaktsituationen sowie bei der Berücksichtigung von Reibung erfordern die Betrachtung von Lösungen aus vorangegangenen Zeitschritten. Echtzeitfähige Berechnung der Kontaktkräfte erzwingt ein Zwischenspeichern der Kräfte, da es sich dabei um gute Startwerte für die Nullstellensuche in der nächsten Iteration handelt.

2.2.2 Der impulsbasierte Ansatz

Die impulsbasierte Simulationsmethode geht auf die Arbeiten von KELLER [Kel86] und HAHN [Hah88] zurück. Entscheidende konzeptionelle Verbesserungen lieferten MIRTICH und CANNY [MC95, Mir95]. Eine vollständige und integrierte Darstellung der einzelnen Beiträge findet sich in der Dissertation von MIRTICH [Mir96b].

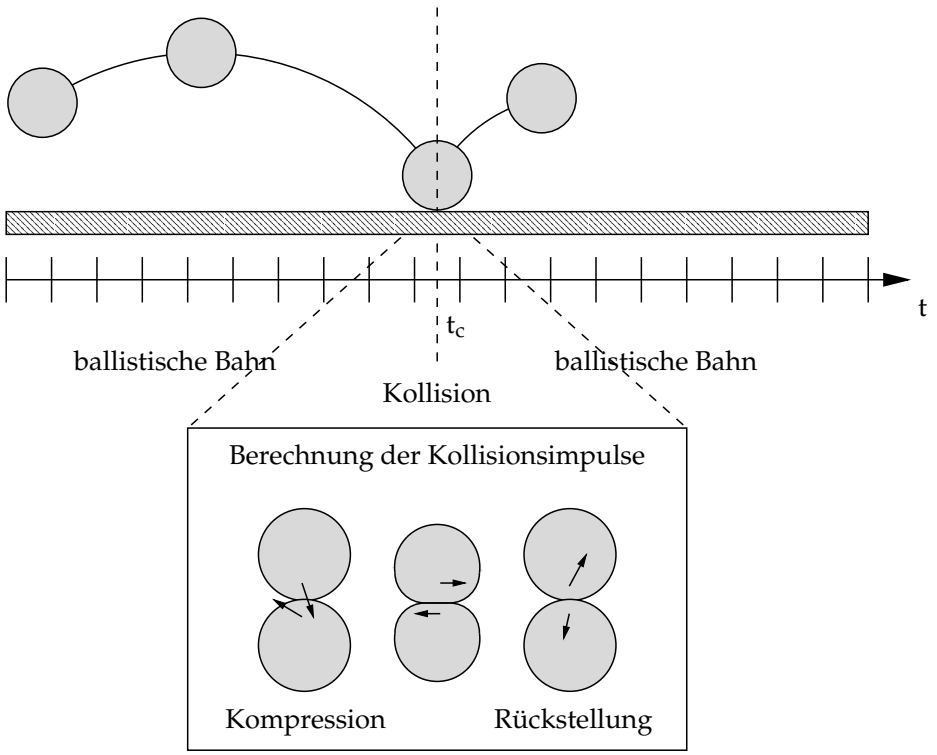
Im Rahmen der impulsbasierten Simulation werden alle Kontaktformen als Kollisionen in einem einzigen Kontaktpunkt (*Einpunktkontakte*) behandelt. Eine Kollision ist ein punktuelles Ereignis auf der Zeitgeraden. Zwischen zwei Kollisionszeitpunkten bewegen sich die Körper auf sogenannten *ballistischen Trajektorien*, d.h. auf kollisionsfreien Bewegungsbahnen, die ausschließlich durch Gravitation beeinflußt werden. Die ballistischen Bewegungsbahnen haben also eine zeitliche Ausdehnung und werden lediglich durch *Kollisionen infinitesimal kleiner Dauer* unterbrochen. Falls ein Körper mehr als einen Kontaktpunkt mit anderen Objekten besitzt, wird diese Situation als eine zeitliche Folge von Einpunktkontakten modelliert. Permanenter Kontakt, wie er zwischen zwei Objekten auftritt, die aufeinander liegen,gleiten oder rollen, kann als eine hochfrequente Folge extrem schwacher Kollisionen, den sogenannten *Mikrokollisionen* aufgefaßt werden. Der impulsbasierte Ansatz verzichtet bewußt auf jede Form von Zwangsbedingungen. Lokale Informationen im Kontaktpunkt bestimmen die Behandlung einer Kollision. Das Paradigma der impulsbasierten Kollision lautet daher:

> Makroskopisch korrektes Verhalten ist das Ergebnis einer präzisen Behandlung der Kollisionen im Kontaktpunkt (mikroskopisches Verhalten).

Eine Kollision wird aufgelöst, indem ein Paar betragsmäßig identischer, jedoch einander entgegengerichteter Impulse berechnet wird. Diese *Kollisionsimpulse* verhindern eine Durchdringung der beiden Körper, falls sie auf die Kollisionspartner angewendet werden. Impulse besitzen dabei die Eigenschaft, die Geschwindigkeiten der Körper unmittelbar zu verändern, so daß sich die kollidierten Körper nach der Kollisionsauflösung wieder auf ballistischen Bahnen bewegen.

Zur Berechnung der Impulse wird die Kollision, welche im impulsbasierten Kontaktmodell eine infinitesimal kleine Dauer besitzt, „unter der Lupe“ einer geeigneten Parametrisierung betrachtet. Unter Berücksichtigung von Reibung und Elastizität der Körper ergibt sich ein Differentialgleichungssystem, das die Änderung der Kollisionsimpulse über die beiden Phasen einer Kollision beschreibt. Entsprechend dem Elastizitätskonzept unterscheidet man die *Kompressionsphase*, in der die Objekte sich zusammendrücken und kinetische Energie in den beiden Körper gespeichert wird, und die *Rückstellungsphase*, in der die Objekte unter Energierückgabe ihre ursprüngliche Form wiedergewinnen. Als Reibungsmodell wird das empirische Modell von COULOMB (vgl. Abschnitt 5.1.3) zugrunde gelegt. Plastizität und Reibung führen zu Energieverlusten innerhalb des Simulationssystems. Energiegewinne sind im Gegensatz zu anderen Modellen, wie beispielsweise dem der Penalty-Methoden aus Abschnitt 2.2.1, nicht möglich, was für die Plausibilität des impulsbasierten Ansatzes spricht. Mit Hilfe der berechneten Kollisionsimpulse können prinzipiell auch Kontaktkräfte bestimmt

Abbildung 2.4 Der impulsbasierte Ansatz zur Dynamiksimulation.



werden, falls diese für haptisches Feedback benötigt werden. Dazu sind lediglich die Kollisionsimpulse innerhalb eines Zeitintervalls aufzusummieren und durch die Länge des Intervalls zu dividieren. Abbildung 2.4 verdeutlicht die Charakteristika des Modells.

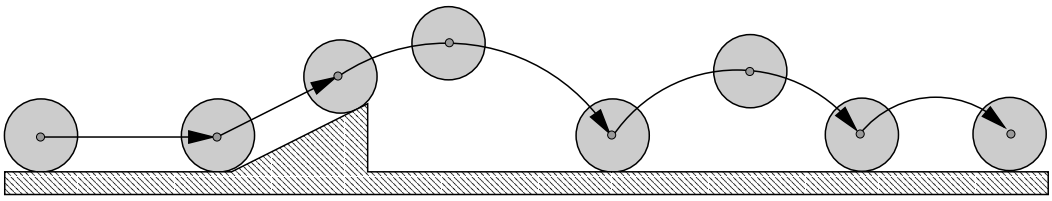
Zur Umsetzung des Ansatzes benötigen wir somit ein Modul zur

- *Bewegungsdurchführung* (Abschnitt 2.5)
  Die Aufgabe der Bewegungsdurchführung ist die Ermittlung und Entwicklung des Dynamikzustandes innerhalb der ballistischen Bewegungsphase aller Körper. Durch Integration der Bewegungsgleichungen (NEWTON-EULER-Differentialgleichungssystem) lassen sich alle kinematischen Daten der Körper während dieses Zeitintervalls bestimmen.

- *Kollisionserkennung* (Kapitel 3 und 4)
  Die Kollisionserkennung bestimmt die Dauer der ballistischen Phase und löst die Kollisionsauflösung aus, sobald zwei Körper miteinander kollidieren.

- *Kollisionsauflösung* (Kapitel 5)
  Das Kollisionsauflösungssystem berechnet im Fall einer Kollision die Kollisionsimpulse und wendet diese auf die Kollisionspartner an.

Abbildung 2.5 Das klassische Beispiel für die Arbeitsweise der impulsbasierten Simulationsmethode.



2.2.3 Vergleich der Ansätze

Um die Unterschiede zum zwangsbasierten Ansatz besser herausstellen zu können, betrachten wir das klassische Beispiel für die Arbeitsweise der impulsbasierten Methode in Abbildung 2.5.

Beispiel 2.3. Eine Kugel rollt eine Rampe hinauf, hebt von dieser ab, springt mehrfach auf die Ebene und rollt schließlich aus. In dem Moment, in dem die Kugel die Rampe verläßt, tritt sie in eine ballistische Bewegungsphase ein. Gravitation ist der einzige externe Einflußfaktor auf die Bewegung, so daß die Kugel schließlich wieder auf der Ebene auftrifft. Die Kollision wird durch die Berechnung und Anwendung der Kollisionsimpulse aufgelöst. Elastizitätseigenschaften beeinflussen die Impulse dabei derart, daß die Kugel nach der Kollision hochspringt und eine neue ballistische Trajektorie bis zum nächsten Kollisionszeitpunkt durchläuft. Energieverluste führen im folgenden zu immer kürzeren und flacheren ballistischen Bahnen. Wenn die Kugel auf der Ebene zu rollen beginnt, sind die kollisionsfreien Phasen derart kurz und die Trajektorien derart flach, daß man die Bewegung als permanenten Kontakt wahrnimmt. □

Im Gegensatz zur zwangsbasierten Simulationsmethode wird zu keinem Zeitpunkt eine Kontaktbedingung formuliert, die die Bewegungsgleichungen im nächsten Zeitschritt einschränkt. Die Einhaltung der *makroskopischen Zwangsbedingung*, der Nichtdurchdringung von Ball und Ebene/Rampe ist das Ergebnis der Auflösung einzelner Kollisionen sowie der Auswertung der Bewegungsgleichungen für die dazwischenliegenden, kollisionsfreien Bewegungsbahnen. Alle Kontaktformen können daher auf ein- und dieselbe Weise behandelt werden. Es ist nicht erforderlich, Unterscheidungen zwischen plastischen/elastischen Stößen sowie Gleit-, Roll- oder Haftvorgängen zu treffen, was selbstverständlich eine Erkennung der jeweiligen Kontaktform auf makroskopischer Ebene voraussetzen würde. Die *uniforme Behandlung aller Kontaktformen* und die daraus resultierende *Einfachheit* des Verfahrens ist gleichzeitig die Schwäche des Ansatzes. Die *große Zahl aufzulösender Kollisionen* im Fall von ruhenden und gleitenden Körpern macht die impulsbasierte Methode in solchen Situationen ineffizient. Hier stellt das zwangsbasierte Modell die natürlichere Beschreibung des Kontaktverhaltens dar. Dies gilt ebenso für den Fall von *Mehrpunktkontakten*, die der impulsbasierte Ansatz nur unbefriedigend behandelt. Ein scheinbar chaotisches Verhalten, wie es beispielsweise eine Menge von Münzen aufweist, die auf eine Ebene fallen,

Tabelle 2.1 Bewertung von zwangsbasiertem und impulsbasiertem Ansatz

Kriterium	*impulsbasierter Ansatz*	*zwangsbasierter Ansatz*
Plastische Stöße	++	++
Elastische Stöße	++	–[1]
Gleitbewegung	+	++
Haftbewegung	○	++
Rollbewegung	–[2]	–[2]
Kohärenzabhängigkeit	++	○
Mehrpunktkontakte	–	++
Physikalische Validität	+	++[3]/○[4]
Implementierbarkeit	++	+
Robustheit	++	+
Parallelisierbarkeit	++	++

ist aufgrund *mangelnder Kohärenz* und *permanent wechselnder Kontaktbeziehungen* durch zwangsbasierte Verfahren kaum zu beschreiben. Hier spielt die impulsbasierte Methode ihre Stärken aus. Es ist offensichtlich, daß die *Vorzüge* der beiden Ansätze *exakt komplementär* liegen. Obwohl es sich um ganzheitliche Konzepte handelt, entscheidet die Anwendung, welches Modell die natürlichere Beschreibung und somit effizientere Simulation ermöglicht. Eines der großen offenen Probleme im Bereich der Simulation starrer Körper ist das Design und die Implementierung eines *hybriden Systems*, das beide Ansätze kombiniert. Unabhängig von diesen inhärenten Vorzügen bzw. Nachteilen der Ansätze spricht für die impulsbasierte Simulation die *einfache Integration des Reibungsmodells sowie des Elastizitätskonzeptes*. Die analytischen zwangsbasierten Ansätze müssen in diesem Zusammenhang Probleme wie beispielsweise die *Unbestimmtheit von Kontaktkräften* oder gar die Frage nach deren *Existenz* lösen. Zudem werden die Modellformulierungen unter Berücksichtigung dieser Erweiterungen komplexer, wie die Linearisierung des Reibungskegels zeigt. Dies hat selbstverständlich Einfluß auf Laufzeit und *Robustheit* der Verfahren. Was die *Parallelisierbarkeit* der Verfahren betrifft, so spricht die *Entkoppelung der Körper* für den impulsbasierten Ansatz. Die Trajektorien der Körper in der ballistischen Bewegungsphase können unabhängig voneinander ermittelt werden. Dies erlaubt, große Zeitschritte durch parallele Berechnungen ohne Interprozessorkommunikation zu überbrücken. Der zwangsbasierte Ansatz betrachtet die Körper nicht derart isoliert, so daß Parallelisierungsmöglickeiten an anderer Stelle gesucht werden müssen. Hier bieten sich in besonderem Maße die Verfahren zu Lösung der LCPs bzw. NCPs an.

Tabelle 2.1 versucht, die Verfahren hinsichtlich der erwähnten Kriterien zu bewer-

[1] Derzeit nicht realisiert, konzeptionell jedoch möglich.
[2] Derzeit vom Reibungsmodell nicht erfaßt, konzeptionell jedoch möglich.
[3] Falls keine Reibung vorliegt.
[4] Existenz und Eindeutigkeit der Kontaktkräfte bei Reibung nicht garantiert.

ten. Sie verzichtet auf Aussagen über das Laufzeitverhalten, da dieses in entscheidendem Maße durch die Anwendung bestimmt wird. In [SSL98] haben wir jedoch gezeigt, daß die Verfahren selbst unter Berücksichtigung von Reibung den gestellten Echtzeitanforderungen genügen.

2.3 Das physikalische und geometrische Modell des starren Körpers

Da wir uns für das dynamische Verhalten realer Objekte interessieren, müssen wir der Simulation ein *physikalisches Körpermodell* zugrunde legen. Physikalisches Verhalten hat seine Ursache im Verhalten auf atomarem Niveau. Dieser Tatsache könnten wir Rechnung tragen, indem wir reale Körper als ein System von einzelnen Massenpunkten auffassen und die Beschreibung des physikalischen Verhaltens auf die Gesetzmäßigkeiten der klassischen Mechanik für einzelne Massenpunkte herunterbrechen. Bei einem makroskopischen Festkörper mit seinen 10^{23} Teilchen pro cm^3 [SSL98] ist ein solches Konzept eher fragwürdig. Da wir nicht an mikroskopischem Verhalten interessiert sind, bietet es sich an, den Körper aus einem *makroskopischen Standpunkt* als ein Ganzes anzusehen. Dies ist eine Idealisierung, die eine mathematische und damit auch rechnergestützte Behandlung der gestellten Probleme erst ermöglicht. In Abschnitt 2.2 haben wir bereits erläutert, daß die Forderung nach einem interaktiven Simulationssystem und den dazu erforderlichen Echtzeitabläufen weitergehende Einschränkungen erzwingen, die sich in dem Modell des *starren Körpers* wiederspiegeln. Aus der physikalischen Betrachtungsweise ist ein starrer Körper ein System von Massenpunkten, deren Abstände im Zeitablauf konstant bleiben [Fli96]. Starre Körper sind somit per (physikalischer) Definition *nicht deformierbar*. Diese Einschränkung erlaubt somit nur solche Objekte zu simulieren, deren Oberflächendeformationen für den Beobachter nicht von Interesse und für das physikalische Verhalten vernachlässigbar sind. In [SSL98] haben wir jedoch festgestellt, daß selbst schwierig zu erklärende, physikalische Phänomene trotz dieser Idealisierung nachvollzogen werden können.

Dem physikalischen Modell des starren Körpers wollen wir einige mathematische Forderungen hinzufügen, die sich unmittelbar aus den Eigenschaften realer Körper ableiten lassen. Aus mathematischer Sicht stellt ein Körper $\mathcal{K}$ zunächst eine Punktmenge dar, die einer zeitlichen Veränderung unterworden ist. Die Zeitabhängigkeit beschreiben wir durch eine Parametrisierung der Ortsvektoren $\mathbf{x}(t)$ aller Punkte $\mathbf{x} \in \mathcal{K}$. Wir können nun eine mathematische Definition des starren Körpers geben:

Definition 2.1 (Starrer Körper). Eine Punktmenge $\mathcal{K} \subset \mathbb{R}^3$ heißt starrer Körper, falls sie die folgenden Eigenschaften erfüllt:

1. $\mathcal{K}$ ist kompakt, d.h. abgeschlossen und beschränkt,
2. $\mathcal{K}$ ist zusammenhängend,
3. $\mathcal{K} = \overline{\text{int}}(\mathcal{K})$,

4. $\forall \mathbf{x}, \mathbf{y} \in \mathcal{K} : \delta\big(\mathbf{x}(t), \mathbf{y}(t)\big) = \text{const.}$

Dabei bezeichnet $\text{int}(\mathcal{K})$ das Innere der Menge $\mathcal{K}$ und $\overline{A} = A \cup \partial A$ den Abschluß der Menge A mit Rand ∂A. □

Forderung 3 stellt dabei sicher, daß der starre Körper nicht zu einer Fläche, einem Liniensegment oder einem Punkt entartet. Im folgenden werden wir die Begriffe Körper, starrer Körper und Objekt synonym verwenden.

Die *Repräsentation des Körpers* hat großen Einfluß auf die Einsatzmöglichkeit und die Effizienz bestimmter Algorithmen. Dies betrifft nicht nur die in dieser Arbeit zu entwickelnden Kollisionserkennungsverfahren, sondern vor allem Algorithmen, die für die Visualisierung der Körper verantwortlich sind. Schnelle Rendering-Algorithmen setzen beispielsweise triangulierte, polygonale Oberflächen voraus [Sch99]. In unserer Simulation wollen wir starre Körper durch *Polyeder* approximieren. Aufgrund planarer Begrenzungsflächen sind sie leicht repräsentierbar und algorithmisch gut zu handhaben. Daneben existieren effiziente Triangulierungsverfahren für Polyederoberflächen als Bestandteil der bekannten Visualisierungbibliotheken. Die Einschränkung auf Polyederapproximationen gefährdet nicht die Güte der Simulationsdaten, da starre Körper beliebig genau angenähert werden können. Wir werden später sehen, daß das Innere des Körpers für unsere Überlegungen von geringem Interesse ist, so daß wir den Körper durch eine Beschreibung seines Randes $\partial\mathcal{K}$ repräsentieren wollen (*boundary representation*). Für Anwendungen, in denen das Körpervolumen eine entscheidende Rolle spielt, bieten sich dagegen volumenbasierte Körperrepräsentationen (*volume representation*) an [NAB86]. Die Randrepräsentation oder *b-rep* umfaßt alle topologischen und geometrischen Informationen über den Rand des Körpers. Die Oberflächenelemente des Körpers $\mathcal{K}$ setzen sich zusammen aus der Menge der Eckpunkte oder *Knoten* $\mathcal{V}(\mathcal{K})$, der *Kantenmenge* $\mathcal{E}(\mathcal{K})$ sowie der *Flächenmenge* $\mathcal{F}(\mathcal{K})$. Zur Beschreibung der *Topologie* müssen die Inzidenzbeziehungen zwischen allen Oberflächenelementen repräsentiert werden. Die *Geometrie* wird dagegen durch Angabe der kartesischen Koordinaten aller Eckpunkte festgelegt. Daneben ist es aus Effizienzgründen erforderlich, Zusatzinformationen über die Oberflächenelemente zu repräsentieren, wie beispielsweise die Flächennormalen. Beim Aufbau der Randrepräsentation müssen einige topologische und geometrische Eigenschaften sichergestellt werden, die unter anderem von Algorithmen zur Bestimmung physikalischer Eigenschaften wie Masse und Trägheitsmomente vorausgesetzt werden:

- *2-Mannigfaltigkeit*:
  Nach [Hof89] ist $\partial\mathcal{K}$ eine 2-Mannigfaltigkeit, wenn die folgende Eigenschaft erfüllt ist:

  $$\begin{aligned} &\forall \mathbf{x} \in \partial\mathcal{K}\ \exists \varepsilon : \ U_\varepsilon(\mathbf{x}) \cap \mathcal{K} \text{ ist topologisch äquivalent zu einer Kreisscheibe} \\ \iff\ &\exists \text{ Bijektion zwischen } U_\varepsilon \cap \mathcal{K} \text{ und der Menge } \left\{\mathbf{y} \in \mathbb{R}^2 \,|\, \|\mathbf{y}\| \leq 1\right\}. \end{aligned}$$

  Im Fall einer Randrepräsentation läßt sich die Bedingung leicht überprüfen: Jede Kante muß inzident zu genau zwei Flächen sein.

- *Abgeschlossenheit*:
  Diese Eigenschaft ist verletzt, wenn es eine Kante gibt, die nur zu einer Fläche inzident ist.

- *Orientierbarkeit*:
  Die Kanten einer Fläche müssen so gerichtet sein, daß bei einem Umlauf des Polygons jede Kante genau einmal besucht wird. Eine Kante muß in den beiden Flächen, zu denen sie inzident ist, entgegengerichtet sein.

Als geometrische Voraussetzung fordern wir, daß alle Flächen planare Polygone darstellen. Dazu müssen wir überprüfen, ob alle Eckpunkte innerhalb der Ebene liegen, die durch den Normalenvektor der Fläche definiert wird. Der Test kann durch Einsetzen der Eckpunkte in die Ebenengleichung absolviert werden. Auf die Beschreibung der b-rep Datenstruktur wollen wir verzichten, da die hier vorgestellten Algorithmen auf einem abstrakteren Niveau beschrieben sind. Einen Überblick über verschiedene Konzepte findet man in [Zac94]. Wir wollen bezüglich der Datenstruktur lediglich annehmen, daß alle inzidenten Kanten eines Knotens sowie alle Kanten einer Fläche in Linearzeit aufgezählt werden können.

2.4 Das geometrische Bewegungsmodell des starren Körpers

Da wir uns zur Aufgabe gestellt haben, das Verhalten von Systemen starrer Körper zu simulieren, müssen wir die Bewegung der Körper im Raum beschreiben können. Dies setzt voraus, daß wir den Ortsvektor jedes Punktes $x \in \mathcal{K}$ zu einem beliebigen Simulationszeitpunkt spezifizieren können. Die Bewegung eines starren Körpers läßt sich somit als eine kontinuierliche Transformation der Punktmenge im Zeitablauf auffassen.

2.4.1 Bewegungen als Spezialfall affiner Abbildungen

Bereits unser Körpermodell verbietet es uns, beliebige Transformationen auf der damit verbundenen Punktmenge vorzunehmen. Darüber hinaus haben wir eine intuitive Vorstellung davon, welche Abbildungen *zulässige Bewegungen* der Körper darstellen. In diesem Abschnitt wollen wir die Transformationen charakterisieren, die unserem Verständnis des Bewegungsbegriffes entsprechen.

Da die Oberflächenelemente der Randrepräsentation Teilmengen *affiner Teilräume* darstellen, dürfen wir nur solche Abbildungen erlauben, die affine Kombinationen unverändert lassen. Dabei handelt es sich um die sogenannten *affinen Abbildungen*.

Definition 2.2. Affine Abbildung Sei $\mathbf{A} \in \mathbb{R}^{3\times3}$, $\mathbf{a} \in \mathbb{R}^3$, dann heißt die Abbildung

$$\begin{aligned} \alpha : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{x} &\mapsto \mathbf{A}\mathbf{x} + \mathbf{a} \end{aligned}$$

eine affine Abbildung des $\mathbb{R}^3$ in sich. □

Die Starrheitseigenschaft der Körper setzt zudem voraus, daß die Transformationen *längenerhaltend* oder *kongruent* sind.

Lemma 2.1 (Kongruenz affiner Abbildungen). *Eine affine Abbildung* $\alpha : \mathbb{R}^3 \to \mathbb{R}^3$, $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{a}$ *ist genau dann kongruent, wenn* A *eine orthonormale Matrix ist, d.h. wenn* $\mathbf{A}^T\mathbf{A} = \mathbf{E}$ *gilt.*

Beweis. α ist kongruent, falls $\|\alpha(\mathbf{x}) - \alpha(\mathbf{y})\| = \|\mathbf{x} - \mathbf{y}\|$ für alle $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ ist. Für zwei beliebige Vektoren $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ gilt nun:

$$\begin{aligned} & \|\alpha(\mathbf{x}) - \alpha(\mathbf{y})\| = \|\mathbf{x} - \mathbf{y}\| \\ \iff \quad & (\mathbf{x} - \mathbf{y})^T\mathbf{A}^T\mathbf{A}(\mathbf{x} - \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y}) \\ \iff \quad & (\mathbf{x} - \mathbf{y})^T(\mathbf{A}^T\mathbf{A} - \mathbf{E})(\mathbf{x} - \mathbf{y}) = \mathbf{0}\,. \end{aligned}$$

Diese Bedingung ist offensichtlich genau dann erfüllt, wenn $\mathbf{A}^T\mathbf{A} = \mathbf{E}$ gilt. □

Aus der Orthonormalitätseigenschaft von $\mathbf{A}$ folgt weiter, daß α *bijektiv* und *winkelerhaltend* ist. Um *Spiegelungen* (*Umlegungen*) auszuschließen, fordern wir schließlich, daß stets $\det(\mathbf{A}) = 1$ gilt. Diese Einschränkungen erlauben die Definition der sogenannten *eigentlichen Bewegungen*.

Definition 2.3 (Eigentliche Bewegungen). Eine kongruente Abbildung $\alpha : \mathbb{R}^3 \to \mathbb{R}^3$, $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{a}$ heißt eigentliche Bewegung, falls $\det(\mathbf{A}) = 1$ gilt. □

Die Euklidische Bewegungsgruppe

Die eigentlichen Bewegungen können entsprechend unserer Intuition hintereinander ausgeführt werden. Das Ergebnis ist wieder eine eigentliche Bewegung.

Definition 2.4 (Hintereinanderausführung affiner Abbildungen). Seien β, γ affine Abbildungen mit

$$\begin{aligned} \beta : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{x} \mapsto \mathbf{B}\mathbf{x} + \mathbf{b}\,, \\ \gamma : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{x} \mapsto \mathbf{C}\mathbf{x} + \mathbf{c}\,. \end{aligned}$$

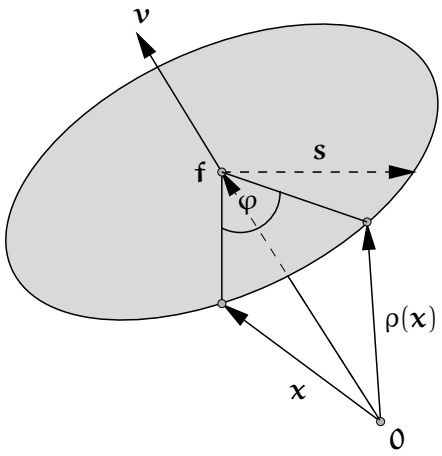
Dann heißt die affine Abbildung $\alpha : \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{a}$, mit $\mathbf{A} := \mathbf{C}\mathbf{B}$ und $\mathbf{a} := \mathbf{C}\mathbf{b} + \mathbf{c}$ die Hintereinanderausführung von β und γ in dieser Reihenfolge. □

Das folgende Lemma ist nun leicht einzusehen.

Lemma 2.2 (Die Euklidische Bewegungsgruppe). *Die Gesamtheit aller eigentlichen Bewegungen bildet bezüglich der Hintereinanderausführung eine Gruppe, die man Euklidische Bewegungsgruppe nennt.*

Falls es nun eine ausreichend kleine Anzahl von eigentlichen Bewegungen gibt, so daß deren Hintereinanderausführung die vollständige Gruppe erzeugt, dann haben wir eine einfache Möglichkeit zur Beschreibung der von uns erlaubten Bewegungen

Abbildung 2.6 Drehung eines Punktes x um eine Ursprungsachse in Richtung des Einheitsvektors $\boldsymbol{\nu}$ um den Winkel φ.



starrer Körper gefunden. Wir wollen zunächst zwei wichtige Vertreter der Euklidischen Bewegungsgruppe vorstellen.

1. Die Translation

Definition 2.5 (Translation). Sei $t \in \mathbb{R}^3$, dann heißt die Abbildung:

$$\begin{aligned} \tau : \mathbb{R}^3 &\to \mathbb{R}^3 \\ \mathbf{x} &\mapsto \mathbf{x} + \mathbf{t} \end{aligned}$$

Translation des $\mathbb{R}^3$ um den Translationsvektor $\mathbf{t}$.

□

2. Die Rotation

Wir wollen zunächst die Abbildung ρ charakterisieren, die einen Punkt $\mathbf{x} \in \mathbb{R}^3$ um eine Ursprungsgerade g mit normiertem Richtungsvektor $\boldsymbol{\nu}$ dreht. Den orientierten Drehwinkel bezeichnen wir mit φ. Dann ist $\mathbf{f} = (\boldsymbol{\nu}^T\mathbf{x})\boldsymbol{\nu} = (\boldsymbol{\nu}\boldsymbol{\nu}^T)\mathbf{x}$ der Lotfußpunkt von $\mathbf{x}$ auf g. Der Vektor $\mathbf{s} = \boldsymbol{\nu} \times (\mathbf{x} - \mathbf{f})$ steht senkrecht auf $\boldsymbol{\nu}$ und $\mathbf{x} - \mathbf{f}$. Aus Abbildung 2.6 ist ersichtlich, daß $\rho(\mathbf{x})$ folgendermaßen dargestellt werden kann:

$$\begin{aligned} \rho(\mathbf{x}) &= \mathbf{f} + (\mathbf{x} - \mathbf{f})\cos\varphi + \mathbf{s}\sin\varphi \\ &= (\boldsymbol{\nu}^T\mathbf{x})\boldsymbol{\nu} + \left[\mathbf{x} - (\boldsymbol{\nu}^T\mathbf{x})\boldsymbol{\nu}\right]\cos\varphi + \boldsymbol{\nu} \times \left[\mathbf{x} - (\boldsymbol{\nu}^T\mathbf{x})\boldsymbol{\nu}\right]\sin\varphi \\ &= \cos\varphi\,\mathbf{x} + (1 - \cos\varphi)(\boldsymbol{\nu}^T\mathbf{x})\boldsymbol{\nu} + \sin\varphi\,(\boldsymbol{\nu} \times \mathbf{x}) \\ &= \cos\varphi\,\mathbf{x} + (1 - \cos\varphi)(\boldsymbol{\nu}\boldsymbol{\nu}^T)\mathbf{x} + \sin\varphi\,(\boldsymbol{\nu} \times \mathbf{x})\,. \end{aligned}$$

Schreiben wir das Kreuzprodukt $\boldsymbol{\nu} \times \boldsymbol{x}$ als Matrix-Vektorprodukt $\mathbf{v}^{\times}x$, so erhalten wir:

$$\begin{aligned} \rho(\boldsymbol{x}) &= \left[\cos\varphi\,\mathbf{E} + (1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} + \sin\varphi\,\mathbf{v}^{\times}\right]\boldsymbol{x} \\ &= \mathbf{R}_{\boldsymbol{\nu},\varphi}\boldsymbol{x}\,, \end{aligned}$$

mit $\mathbf{R}_{\boldsymbol{\nu},\varphi} := \cos\varphi\,\mathbf{E} + (1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} + \sin\varphi\,\mathbf{v}^{\times}$.
Diese Gleichung wird auch als Formel von RODRIGUES bezeichnet.

Definition 2.6 (Rotation). Die Abbildung

$$\begin{aligned} \rho : \mathbb{R}^3 &\to \mathbb{R}^3 \\ \boldsymbol{x} &\mapsto \mathbf{R}_{\boldsymbol{\nu},\varphi}\boldsymbol{x} \end{aligned}$$

wird als *Rotation* mit Drehwinkel φ um die durch den normierten Richtungsvektor $\boldsymbol{\nu}$ definierte Ursprungsgerade bezeichnet. □

Lemma 2.3. *Bei den in 2.5 bzw. 2.6 definierten Abbildungen Translation und Rotation handelt es sich um eigentliche Bewegungen.*

Beweis. Translation und Rotation sind offensichtlich affine Abbildungen. Um dies einzusehen, setzt man im Fall der Translation $\mathbf{A} := \mathbf{E}$ und $\mathfrak{a} := \mathbf{t}$ bzw. für die Rotation $\mathbf{A} := \mathbf{R}_{\boldsymbol{\nu},\varphi}$ und $\mathfrak{a} := 0$. Die Kongruenzeigenschaft ist im Fall der Translation trivialerweise erfüllt, da $\mathbf{E}^{\mathrm{T}}\mathbf{E} = \mathbf{E}$ gilt. Auch die Rotation ist längenerhaltend, wie man leicht einsieht:

$$\begin{aligned} \mathbf{R}^{\mathrm{T}}_{\boldsymbol{\nu},\varphi}\mathbf{R}_{\boldsymbol{\nu},\varphi} &= \left[\cos\varphi\,\mathbf{E} + (1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} - \sin\varphi\,\mathbf{v}^{\times}\right]^{\mathrm{T}} \\ &\quad \cdot\left[\cos\varphi\,\mathbf{E} + (1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} + \sin\varphi\,\mathbf{v}^{\times}\right] \\ &= \left[\cos\varphi\,\mathbf{E} + (1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}}\right]^2 - (\sin\varphi\,\mathbf{v}^{\times})^2 \\ &= \cos^2\varphi\,\mathbf{E} + (1-\cos\varphi)^2\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} + 2\cos\varphi(1-\cos\varphi)\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} - \sin^2\varphi\,(\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} - \mathbf{E}) \\ &= \mathbf{E}\,. \end{aligned}$$

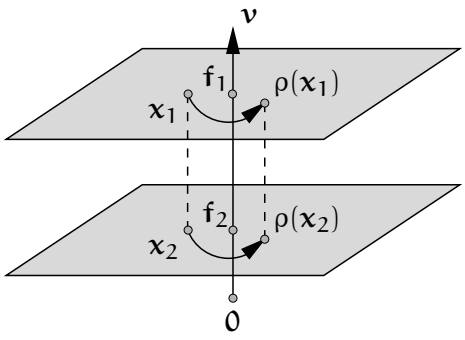
Dabei haben wir verwendet, daß $(\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}})^2 = \boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}}$, $\mathbf{v}^{\times}(\boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}}) = 0$ und $\mathbf{v}^{\times}\mathbf{v}^{\times} = \boldsymbol{\nu}\boldsymbol{\nu}^{\mathrm{T}} - \mathbf{E}$ gilt.
Die Auswertung der Determinante von $\mathbf{R}_{\boldsymbol{\nu},\varphi}$ liefert schließlich $\det(\mathbf{R}_{\boldsymbol{\nu},\varphi}) = \det(\mathbf{E}) = 1$, was unsere Behauptung beweist. □

2.4.2 Die Darstellung eigentlicher Bewegungen durch Translation und Rotation

Wir wollen im folgenden zeigen, daß die vollständige Euklidische Bewegungsgruppe durch die beiden Vertreter Translation und Rotation erzeugt werden kann.

Definition 2.7 (Fixpunkt und nullfixe Abbildung). Ein Punkt $\boldsymbol{x}$ heißt Fixpunkt einer affinen Abbildung $\alpha : \mathbb{R}^3 \to \mathbb{R}^3$, $\boldsymbol{x} \mapsto \mathbf{A}\boldsymbol{x} + \mathfrak{a}$, wenn $\boldsymbol{x} = \mathbf{A}\boldsymbol{x} + \mathfrak{a}$ gilt.
Ist der Fixpunkt der Koordinatenursprung, d.h. $\boldsymbol{x} = \mathbf{A}\boldsymbol{x}$, dann heißt die affine Abbildung nullfix. □

Abbildung 2.7 Fall 1: Lösungsmenge ist Fixgerade durch den Ursprung mit Richtungsvektor ν.



Die Definition erlaubt die Formulierung eines leicht einzusehenden Darstellungssatzes für affine Abbildungen.

Lemma 2.4 (Darstellung affiner Abbildungen). *Jede affine Abbildung ist die Hintereinanderausführung einer nullfixen Abbildung und einer Translation.*

Der folgende Satz liefert schließlich das gesuchte Resultat.

Satz 2.5. *Eine nullfixe eigentliche Bewegung ist eine Rotation um eine Ursprungsgerade.*

Beweis. Die Fixpunkte der nullfixen eigentlichen Bewegung α erfüllen das lineare Gleichungssystem $(\mathbf{A}-\mathbf{E})\mathbf{x}=0$.
1.Fall: $\operatorname{rang}(\mathbf{A}-\mathbf{E})=2$

In diesem Fall stellt die Lösungsmenge eine Fixgerade g durch den Ursprung mit Richtungsvektor ν dar. Wir betrachten zwei Ebenen Σ_1 und Σ_2, die g in den Punkten f_1 und f_2 senkrecht schneiden. Abbildung 2.7 veranschaulicht die Situation. Sei $x_1 \in \Sigma_1$ mit $x_1 \neq f_1$. Da $\alpha(f_1)=f_1$ und $\alpha(f_2)=f_2$ gilt und α winkelerhaltend ist folgt $\sphericalangle(x_1,f_1,f_2)=\sphericalangle(\alpha(x_1),f_1,f_2)$ und somit $\alpha(x_1)\in\Sigma_1$. Aufgrund der Längenerhaltung von α gilt weiter $\|\alpha(x_1-f_1)\|=\|x_1-f_1\|$. Die Einschränkung von α auf Σ_1 ist offensichtlich eine Rotation um g durch einen Winkel φ_1. Analog sehen wir, daß die Einschränkung von α auf Σ_2 eine Rotation um den Drehwinkel φ_2 darstellt.

Es bleibt zu zeigen, daß $\varphi_1=\varphi_2$ gilt. Dazu nehmen wir an, daß x_2 die Projektion parallel zu g auf Σ_2 ist. Die Kongruenzeigenschaft von α liefert nun: $\|x_1-f_1\| = \|x_2-f_2\| = \|\alpha(x_1)-f_1\| = \|\alpha(x_2)-f_2\|$ und $\|x_1-x_2\| = \|\alpha(x_1)-\alpha(x_2)\|$. Aufgrund der Parallelität von Σ_1 und Σ_2 gilt schließlich: $\|x_1-\alpha(x_1)\| = \|x_2-\alpha(x_2)\|$. Somit sind die Dreiecke $\Delta(x_1,f_1,\alpha(x_1))$ und $\Delta(x_2,f_2,\alpha(x_2)$ kongruent und $\varphi_1=\varphi_2$.

2.Fall $\operatorname{rang}(\mathbf{A}-\mathbf{E})=1$
Die Lösungsmenge beschreibt eine Fixebene $\Sigma=\{x\in\mathbb{R}^3 \mid x=\lambda\nu+\mu w, \nu, w\in\mathbb{R}^3, \lambda\in\mathbb{R}, \|\nu\|=\|w\|=1\}$. Sei $x\notin\Sigma$ und f die senkrechte Projektion von x auf Σ. Aufgrund

der Winkel- und Längenerhaltung von α gilt, daß $\alpha(\mathbf{x}) - \mathbf{f}$ senkrecht auf Σ steht und $\|\mathbf{x} - \mathbf{f}\| = \|\alpha(\mathbf{x}) - \mathbf{f}\|$. Daher ist α entweder eine Spiegelung an Σ oder die Identität. Da eine Spiegelung keine eigentliche Bewegung darstellt, folgt unsere Behauptung auch für diesen Fall.

3.Fall $\quad \operatorname{rang}(\mathbf{A} - \mathbf{E}) = 0$
In diesem Fall gilt $\mathbf{A} = \mathbf{E}$. Die nullfixe eigentliche Bewegung ist also die Identität und somit eine Rotation. □

Korollar 2.6. Jede von uns erlaubte Bewegung eines starren Körpers kann durch die Hintereinanderausführung einer Rotation um eine Ursprungsgerade und einer anschließenden Translation dargestellt werden.

2.5 Das physikalische Bewegungsmodell des starren Körpers

In diesem Abschnitt werden wir die Bewegung des starren Körpers aus der physikalischen Perspektive betrachten und dabei sehen, daß die geometrische und physikalische Bewegungsbeschreibung äquivalent sind. Ausgehend von der geometrischen Definition des starren Körpers und einigen intuitiven Vorstellungen hinsichtlich des Bewegungsbegriffes haben wir im vorangegangenen Abschnitt eine Teilmenge der affinen Abbildungen, die sogenannten *eigentlichen Bewegungen*, als zulässige Bewegungstransformationen identifiziert. Das Teilgebiet der Mechanik, das sich mit der Beschreibung der Körperbewegungen beschäftigt, ist die sogenannte *Kinematik*. Sie grenzt sich auf diese Weise von der *Dynamik* ab, die nach den physikalischen Gesetzen fragt, die der Körperbewegung zugrundeliegen. Wir wollen in diesem Abschnitt eine Einführung in beide Teilgebiete geben, mit dem Ziel der Herleitung eines Differentialgleichungssystems zur Beschreibung der Bewegung starrer Körper unter dem Einfluß von Kräften. Diese Fragestellung ist, wie wir bereits in Abschnitt 2.4 angemerkt haben, gut verstanden und somit Gegenstand der Standardliteratur zur klassischen Mechanik. Wir wollen uns hier an einer Übersichtsdarstellung von BARRAFF [Bar97] orientieren, die die relevanten physikalischen Größen und ihre Beziehungen vorstellt und dabei die Analogie von translatorischer und rotatorischer Bewegungskomponente des starren Körpers herausstellt. Dabei beginnen wir mit der Kinematik, die die Bewegung starrer Körper im Rahmen der Theorie relativ zueinander bewegter Bezugssysteme beschreibt.

2.5.1 Position und Orientierung

Bewegung ist ein relativer Begriff, so daß es Sinn macht, Bewegungen relativ zu einem *Bezugssystem* wie beispielsweise dem Raum, in dem wir uns befinden, zu definieren. Durch Einführung von Ortskoordinaten entsteht aus dem Bezugssystem ein *Koordinatensystem*. Koordinatensysteme spielen daher in der Bewegungsbeschreibung eine entscheidende Rolle.

Definition 2.8 (Koordinatensystem). Sei $A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ eine Orthonormalbasis des $\mathbb{R}^3$ mit $\det(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) = 1$ und sei $\mathbf{o} \in \mathbb{R}^3$. Dann bezeichnen wir $(A, \mathbf{o})$ als Koordi-

natensystem mit Ursprung $\mathbf{o}$ und Koordinatenrichtungen $\mathbf{a}_1$, $\mathbf{a}_2$ und $\mathbf{a}_3$. Die Matrix $\mathbf{R} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ heißt Orientierungsmatrix von $(A, \mathbf{o})$. □

Zur Beschreibung der Bewegung eines starren Körpers führen wir das sogenannte *Weltkoordinatensystem* $\mathcal{O} = (E, \mathbf{0})$ ein, dessen Koordinatenrichtungen $E = \{e_1, e_2, e_3\}$ die Standardbasis des $\mathbb{R}^3$ bilden und dessen Ursprung durch den Nullvektor beschrieben wird. Daneben betrachten wir das *Körperkoordinatensystem* $\mathcal{B} = (B, \mathbf{c})$ mit Orthonormalbasis $B = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$, das fest im *Schwerpunkt* c des Körpers verankert ist und sich relativ zum raumfesten Weltkoordinatensystem bewegt. Den Begriff des Schwerpunktes werden wir später definieren. Es genügt an dieser Stelle, ihn sich als ausgezeichneten Punkt im geometrischen Zentrum des Körpers vorzustellen. Die Orientierungsmatrix $\mathcal{R} = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]$ drückt somit die Basisvektoren von $\mathcal{B}$ in Weltkoordinaten aus. Im folgenden werden wir jeden Vektor bezüglich $\mathcal{O}$ interpretieren und explizit das Subskript $\mathcal{B}$ verwenden, wenn wir den Vektor im Körperkoordinatensystem ausdrücken wollen.

Aufgrund dieser Vereinbarungen kann jeder in Körperkoordinaten ausgedrückte Punkt $x_{\mathcal{B}}$ einem Punkt x im Weltkoordinatensystem entsprechend der folgenden Beziehung zugeordnet werden:

$$\mathbf{x} = \mathbf{R}x_{\mathcal{B}} + \mathbf{c} \,.$$

Diese Abbildung stellt offensichtlich eine *eigentliche Bewegung* entsprechend Definition 2.3 dar und kann als starre Transformation des Weltkoordinatensystems in das Körperkoordinatensystem interpretiert werden: $\mathbf{R} = \mathbf{R}_{\boldsymbol{\nu},\varphi}$ beschreibt dabei eine Rotation mit Drehwinkel φ des Weltkoordinatensystems um eine Ursprungsachse $\boldsymbol{\nu}$ und $\mathbf{c}$ die anschließende Translation des rotierten Systems. Selbstverständlich sind $\mathbf{R}$ und $\mathbf{c}$ im Rahmen der Bewegungsbeschreibung zeitabhängig. Wir wollen jedoch aus Gründen der Übersichtlichkeit den Zeitparameter t nicht explizit mitführen. Den Schwerpunktvektor $\mathbf{c}$ bezeichnen wir als *Position* des Körpers $\mathcal{K}$ und $\mathbf{R}$ als dessen *Orientierung*. Die Lage des starren Körpers kann also durch Angabe seiner Position und Orientierung zu jedem Zeitpunkt der Simulation spezifiziert werden.

2.5.2 Lineare Geschwindigkeit und Winkelgeschwindigkeit

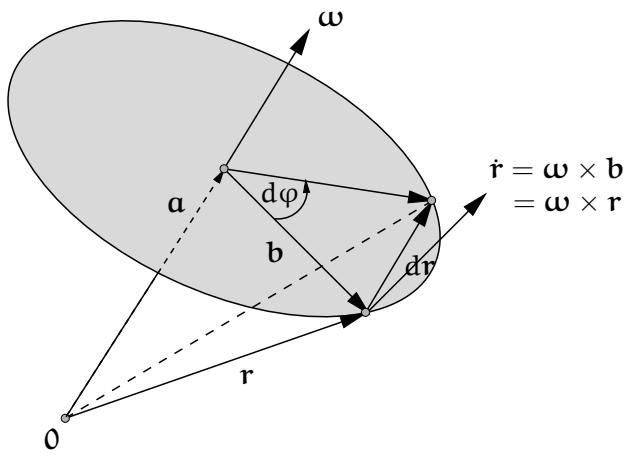
Unser Ziel besteht darin, die Bewegung des Körpers über die Simulationsdauer hinweg zu beschreiben. Wir interessieren uns daher für die Veränderung von Position und Orientierung im Zeitablauf. Unter der *linearen Geschwindigkeit* $\boldsymbol{\nu}$ versteht man die zeitliche Änderung $\dot{\mathbf{c}}$ des Ortsvektors des Massenschwerpunktes:

$$\dot{c} = v = \frac{d}{dt}\,\mathbf{c} \,. \tag{2.3}$$

Das rotatorische Analogon zur linearen Geschwindigkeit ist die *Winkelgeschwindigkeit* des Körpers. Sie gibt als skalare Größe ω die zeitliche Winkeländerung $\dot{\varphi}$ der durch $\mathbf{R} = \mathbf{R}_{\boldsymbol{\nu},\varphi}$ beschriebenen Rotation um eine Schwerpunktachse an:

$$\omega = \dot{\varphi} = \frac{d}{dt}\,\varphi \,.$$

Abbildung 2.8 Die Änderung eines Vektors $\mathbf{r}$ aufgrund einer infinitesimalen Drehung.



Im allgemeinen ist es sinnvoll, der Winkelgeschwindigkeit einen Vektor $\boldsymbol{\omega}$ in Richtung der Drehachse zuzuordnen, dessen Länge gerade ihrem Betrag entspricht. Von nun an wollen wir die Winkelgeschwindigkeit als diesen Vektor auffassen. Da unser Interesse der Zeitableitung der Rotationsmatrix gilt, werden wir im folgenden eine Beziehung zwischen $\boldsymbol{\omega}$ und $\dot{\mathbf{R}}$ herleiten. Dazu betrachten wir die Situation in Abbildung 2.8 . Sei $\mathbf{r}$ ein in Weltkoordinaten ausgedrückter Vektor, der von $\mathcal{B}$ zeitunabhängig ist, da er sich mit dem Körper bewegt. Der Richtungsvektor $\mathbf{r}$ läßt sich somit als Summe der Vektoren $\mathbf{a}$ und $\mathbf{b}$ darstellen, wobei $\mathbf{a}$ parallel und $\mathbf{b}$ orthogonal zum Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}$ ist. Wie aus der Abbildung ersichtlich wird, läuft $\mathbf{r}$ auf einem Kreis mit Radius $\|\mathbf{b}\|$ um die $\boldsymbol{\omega}$-Achse, so daß die zeitliche Änderung von $\mathbf{r}$ senkrecht auf $\mathbf{b}$ und $\boldsymbol{\omega}$ steht. Der Betrag der Momentangeschwindigkeit $\dot{\mathbf{r}}$ eines Vektors $\mathbf{r}$, der bei konstanter Winkelgeschwindigkeit einen Kreis mit Radius $\|\mathbf{b}\|$ durchläuft, beträgt $\|b\|\|\boldsymbol{\omega}\|$. Da $\mathbf{b}$ und $\boldsymbol{\omega}$ orthogonal sind, gilt zudem $\|\boldsymbol{\omega} \times \mathbf{b}\| = \|\boldsymbol{\omega}\|\|\mathbf{b}\|$. Somit folgt

$$\dot{\mathbf{r}} = \boldsymbol{\omega} \times \mathbf{b}$$

und mit $\mathbf{a} \parallel \boldsymbol{\omega}$ schließlich

$$\begin{aligned} \dot{\mathbf{r}} &= \boldsymbol{\omega} \times \mathbf{a} + \boldsymbol{\omega} \times \mathbf{b} = \boldsymbol{\omega} \times (\mathbf{a} + \mathbf{b}) \\ &= \boldsymbol{\omega} \times \mathbf{r} \,. \end{aligned}$$

Diese Eigenschaft gilt insbesondere für die Basisvektoren $\mathrm{B} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ des Körperkoordinatensystems, so daß sich die Spalten der Matrix $\dot{\mathbf{R}}$ als deren Zeitableitungen ergeben:

$$\begin{aligned} \dot{\mathbf{R}} &= [\dot{\mathbf{b}}_1, \dot{\mathbf{b}}_2, \dot{\mathbf{b}}_3] \quad \text{mit} \\ \dot{\mathbf{b}}_i &= \boldsymbol{\omega} \times \mathbf{b}_i \,, \quad 1 \leq i \leq 3 \,. \end{aligned}$$

Verwenden wir die Matrixschreibweise für die Vektorproduktbildung, so erhalten wir die gesuchte Beziehung zwischen der Winkelgeschwindigkeit und der Zeitableitung der Rotationsmatrix:

$$\dot{\mathbf{R}} = \boldsymbol{\omega}^{\times}\mathbf{R}\,. \tag{2.4}$$

Aufgrund dieser Überlegungen können wir nun die *(absolute) Geschwindigkeit* $\mathbf{u} := \dot{\mathbf{x}}$ eines Punktes $x \in \mathcal{K}$ mit $\dot{\mathbf{x}}_{\mathcal{B}} = \mathbf{0}$ angeben:

$$\begin{aligned} \mathbf{u} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{R}\mathbf{x}_{\mathcal{B}} + \mathbf{c}\right) = \dot{\mathbf{R}}\mathbf{x}_{\mathcal{B}} + \mathbf{R}\dot{\mathbf{x}}_{\mathcal{B}} + \dot{\mathbf{c}} \\ &= \boldsymbol{\omega}^{\times}\mathbf{R}\mathbf{x}_{\mathcal{B}} + \boldsymbol{\nu} = \boldsymbol{\omega}^{\times}(\mathbf{x} - \mathbf{c}) + \boldsymbol{\nu} \\ &= \boldsymbol{\omega} \times \mathbf{r} + \boldsymbol{\nu}\,, \end{aligned} \tag{2.5}$$

wobei $\mathbf{r} := \mathbf{x} - \mathbf{c}$ die Verschiebung des Punktes $x \in \mathcal{K}$ gegenüber dem Schwerpunkt beschreibt.

2.5.3 Physikalische Momente des starren Körpers

Wir haben uns bisher darauf beschränkt, die Bewegung des Körpers zu beschreiben, ohne dabei nach den Ursachen dieser Bewegung zu fragen. Im folgenden werden wir die sogenannten *physikalischen Momente* des Körpers definieren, die zur Untersuchung der Wirkung von Kräften und Impulsen auf die Bewegung des starren Körpers unerläßlich sind. Als physikalische Momente bezeichnen wir die *Masse*, den *Schwerpunkt* und die *Trägheitsmatrix* des Körpers.

Stellen wir uns den starren Körper als ein System von Massenteilchen vor, so entspricht seine Gesamtmasse der Summe der einzelnen Partikelmassen. Der Schwerpunkt ist der mit der jeweiligen Partikelmasse gewichtete Mittelwert aller Massenpunkte. Eine Verallgemeinerung des Massenbegriffs stellt die Trägheitsmatrix dar, die die Massenverteilung innerhalb des Körpers beschreibt. Da der starre Körper jedoch kein System endlich vieler Massenpunkte, sondern vielmehr ein Punktekontinuum darstellt, werden seine physikalischen Momente über Volumenintegrale und eine Dichtefunktion definiert.

$$\text{Masse} \;:\quad m := \iiint_{\mathcal{K}} \rho(\mathbf{x})\,\mathrm{d}(x_1, x_2, x_3)\,, \tag{2.6}$$

$$\text{Schwerpunkt} \;:\quad \mathbf{c} := \iiint_{\mathcal{K}} \rho(\mathbf{x})\mathbf{x}\,\mathrm{d}(x_1, x_2, x_3)\,, \tag{2.7}$$

$$\text{Trägheitsmatrix} \;:\quad \mathbf{I} := \iiint_{\mathcal{K}} \rho(\mathbf{r})(\mathbf{r}^2\mathbf{E} - \mathbf{r}\mathbf{r}^{\mathrm{T}})\,\mathrm{d}(x_1, x_2, x_3)\,. \tag{2.8}$$

Dabei bezeichnet $\mathbf{r} = \mathbf{x} - \mathbf{c}$ die Verschiebung des Punktes $x \in \mathcal{K}$ gegenüber dem Schwerpunkt und $\rho(\mathbf{x})$ die Massendichte des Körpers im Punkt x. In unserem Modell wollen wir annehmen, daß die Masse *homogen* über den Körper verteilt ist, d.h. daß die Massendichte in allen Punkten des Körpers gleich ist. Es gilt also $\rho(\mathbf{x}) = \rho$ für alle $x \in \mathcal{K}$.

Mit Hilfe der Integralsätze ist es nun möglich, die auftretenden Volumenintegrale auf Flächenintegrale und schließlich Linienintegrale zurückzuführen, die dann geschlossen oder numerisch gelöst werden müssen. Eine detaillierte Beschreibung der Vorgehensweise sowie einen Algorithmus für die von uns verwendete Polyederdarstellung des Körpers findet sich in der Arbeit von MIRTICH [Mir96a].

Die Trägheitsmatrix ist in 2.8 bezüglich des Schwerpunktes und in Weltkoordinaten ausgedrückt. In einem vorgegebenen Koordinatensystem sind die Komponenten der Matrix eindeutig durch die Massenverteilung des starren Körpers bestimmt. Dies bedeutet, daß sich bei einer Rotation des Körpers die Komponenten von $\mathbf{I}$ ändern. Die Trägheitsmatrix ist also, im Weltkoordinatensystem ausgedrückt, *zeitabhängig*. Bestimmen wir $\mathbf{I}$ dagegen bezüglich des Körperkoordinatensytems, so erhalten wir eine zeitlich invariante Matrix $\mathbf{I}_{\mathcal{B}}$. Die Frage, wie die beiden Trägheitsmatrizen $\mathbf{I}$ und $\mathbf{I}_{\mathcal{B}}$ zusammenhängen, beantwortet der Satz von STEINER [Nol96]. Die von uns gesuchte Beziehung ist ein Spezialfall einer allgemeineren Aussage über das Transformationsverhalten der Trägheitsmatrix und kann folgendermaßen formuliert werden:

$$\mathbf{I} = \mathbf{R}\mathbf{I}_{\mathcal{B}}\mathbf{R}^{T} . \tag{2.9}$$

Neben dieser Transformationseigenschaft sieht man leicht ein, daß $\mathbf{I}$ und somit auch $\mathbf{I}_{\mathcal{B}}$ *symmetrisch* und *reell* sind. Es läßt sich nun zeigen, daß bei festem Koordinatenursprung eine Drehung des Bezugssystems existiert, die die Nichtdiagonalelemente zum Verschwinden bringt. Diese Drehung bezeichnet man als *Hauptachsentransformation*. Die so festgelegten Koordinatenachsen heißen *Hauptträgheitsachsen* und die Diagonalelemente der entsprechenden Matrix *Hauptträgheitsmomente*. Da wir uns in der Wahl der Körperkoordinatenachsen nicht festgelegt haben, bietet es sich an, diese Freiheit zu nutzen und die normierten Hauptträgheitsachsen als Basisvektoren des Körperkoordinatensystems zu verwenden.

2.5.4 Kraft und Drehmoment

Die *Kraft* ist die zentrale physikalische Größe, die für die Änderung eines Bewegungszustandes ursächlich ist. Sie wird im allgemeinen durch den Buchstaben $\mathbf{F}$ symbolisiert. Im Rahmen der *dynamischen Grundgleichung* (2. Newtonsches Axiom) wird der Kraftbegriff folgendermaßen definiert:

$$\mathbf{F} = \frac{d}{dt}\left(m\dot{\mathbf{x}}\right) . \tag{2.10}$$

Da wir keine relativistische Mechanik betreiben wollen und somit die Masse als zeitunabhängig betrachten, gilt:

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

wobei wir $\mathbf{a} = \ddot{\mathbf{x}}$ als *Beschleunigungsvektor* bezeichnen.

Wenn wir von einer Kraft reden, die aufgrund externer Einflüsse wie beispielsweise Gravitation auf einen starren Körper wirkt, dann stellen wir uns vor, daß diese Kraft an einem bestimmten Punkt des Körpers angreift. Die Gesamtkraft $\mathbf{F}$, die auf einen

solchen Punkt $x \in \mathcal{K}$ wirkt, ergibt sich entsprechend dem *Superpositionsprinzip* als Summe der Einzelkräfte, die an diesem Punkt angreifen. Sie induziert in dem Punkt x ein sogenanntes *Drehmoment* $\mathbf{d}$, das folgendermaßen definiert ist:

$$\mathbf{d} = (\mathbf{x} - \mathbf{c}) \times \mathbf{F} = \mathbf{r} \times \mathbf{F} .$$

Das Drehmoment ist also im Gegensatz zur Kraft abhängig von der relativen Lage des Angriffspunktes zum Massenschwerpunkt. Die Richtung von $\mathbf{d}$ kann man sich als die Schwerpunktachse vorstellen, um die der Körper sich aufgrund der nicht im Schwerpunkt angreifenden Kraft dreht (Schwerpunkt bleibt stabil). Wir nehmen nun an, daß mehrere Kräfte $\mathbf{F}_1, \ldots, \mathbf{F}_n$, auf den Körper wirken. Ferner sei $\mathbf{r}_i$, $1 \leq i \leq n$, der Verschiebungsvektor des i-ten Angriffspunktes x_i gegenüber dem Schwerpunkt. Dann ergibt sich die Gesamtkraft $\mathbf{F}$ sowie das Gesamtdrehmoment $\mathbf{D}$ des Körpers als Summe der an den einzelnen Angriffpunkten wirkenden Kräfte bzw. der dort induzierten Drehmomente:

$$\mathbf{F} = \sum_{i=1}^{n} \mathbf{F}_i \tag{2.11}$$

$$\mathbf{D} = \sum_{i=1}^{n} \mathbf{d}_i = \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{F}_i \tag{2.12}$$

2.5.5 Linearer Impuls und Drehimpuls

Im Rahmen des impulsbasierten Kollisionsmodells betrachten wir statt der auf den Körper wirkenden Gesamtkraft die Änderung des Gesamtimpulses. Zum besseren Verständnis des impulsbasierten Simulationskonzepts wollen wir an dieser Stelle die Zusammenhänge zwischen linearem Impuls und Kraft sowie zwischen Drehimpuls und Drehmoment vorstellen. Wir definieren zunächst den *linearen Impuls* eines einzelnen Massenpunktes x mit Masse m und Geschwindigkeit $\dot{\mathbf{x}} = \boldsymbol{\nu}$ als:

$$p = m\dot{\mathbf{x}} = m\boldsymbol{\nu} .$$

Betrachten wir den starren Körper vereinfacht als n-Teilchensystem, so können wir den linearen Impuls des Körpers als Summe der linearen Impulse der einzelnen Massenpunkte auffassen:

$$\mathcal{P} = \sum_{i=1}^{n} m_i \dot{\mathbf{x}}_i ,$$

wobei m_i die Masse des Punktes x_i, $1 \leq i \leq n$, bezeichnet. Im Fall des starren Körpers haben wir es jedoch mit einem Punktekontinuum zu tun, so daß wir statt der Summation über alle Massenpunkte auf die Volumenintegration zurückgreifen müssen:

$$\mathcal{P} = \int\int\int_{\mathcal{K}} \rho \dot{\mathbf{x}} \, d(x_1, x_2, x_3) .$$

Verwenden wir Gleichung 2.5 für die Geschwindigkeit eines Massenpunktes, so erhalten wir:

$$\begin{aligned}\mathcal{P} &= \iiint_{\mathcal{K}} \rho\dot{\mathbf{x}}\, d(x_1, x_2, x_3) \\ &= \iiint_{\mathcal{K}} \rho(\boldsymbol{\nu} + \boldsymbol{\omega} \times \mathbf{r})\, d(x_1, x_2, x_3) \\ &= \iiint_{\mathcal{K}} \rho\boldsymbol{\nu}\, d(x_1, x_2, x_3) \;+\; \boldsymbol{\omega} \times \iiint_{\mathcal{K}} \rho\mathbf{r}\, d(x_1, x_2, x_3)\,.\end{aligned}$$

Da der Ursprung des Körperkoordinatensystems im Massenschwerpunkt liegt, gilt:

$$\iiint_{\mathcal{K}} \rho\mathbf{r}\, d(x_1, x_2, x_3) = \mathbf{0}\,.$$

Der lineare Gesamtimpuls des Körpers lautet also:

$$\mathcal{P} = m\dot{\mathbf{c}} = m\boldsymbol{\nu}\,.$$

Differenziert man diese Gleichung, so erhält man die dynamische Grundgleichung

$$\dot{\mathcal{P}} = m\ddot{\mathbf{c}} = m\mathbf{a} = \mathbf{F}\,.$$

Diese Gleichung besagt, daß die Änderung des linearen Impulses äquivalent zu der Gesamtkraft ist, die auf den Körper wirkt.

Das Konzept des Drehimpulses ist weniger intuitiv als das des linearen Impulses, erlaubt es jedoch, viele Gleichungen einfacher zu formulieren. Analog zur Vorgehensweise im Fall des Drehmomentes definieren wir den *Drehimpuls*, der in einem Punkt x des Körpers induziert wird als:

$$\mathbf{l} = (\mathbf{x} - \mathbf{c}) \times \mathbf{p} = \mathbf{r} \times \mathbf{p}\,.$$

Der *Gesamtdrehimpuls* des Körpers ergibt sich erneut durch Integration über das Volumen des starren Körpers:

$$\mathbf{L} = \iiint_{\mathcal{K}} \mathbf{r} \times \mathbf{p}\, d(x_1, x_2, x_3)\,.$$

Man kann nun leicht zeigen, daß der Drehimpuls eine lineare Funktion der Winkelgeschwindigkeit ist. Es gilt nämlich:

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega}\,. \tag{2.13}$$

Wie im Fall des linearen Impulses liefert die Zeitableitung von $\mathbf{L}$ die gesuchte Beziehung zwischen Drehimpuls und Drehmoment:

$$\dot{\mathbf{L}} = \frac{d}{dt}\mathbf{L} = \mathbf{D}\,.$$

Diese in der Mechanik als Drehimpulssatz bekannte Gleichung rechtfertigt im impulsbasierten Kollisionskonzept die Berechnung der Drehimpulsänderung zur Bestimmung des Drehmomentes.

Aus dem Drehimpulssatz wollen wir nun die EULER-Gleichungen ableiten. Dabei handelt es sich um ein gekoppeltes Differentialgleichungssystem für die Komponenten der Winkelgeschwindigkeit.

$$\begin{aligned}\mathbf{D} &= \frac{d}{dt}\left(\mathbf{I}\omega\right) = \frac{d}{dt}\left(\mathbf{R}\mathbf{I}_{\mathcal{B}}\mathbf{R}^{T}\right)\omega + \mathbf{I}\dot{\omega} \\ &= \left(\dot{\mathbf{R}}\mathbf{I}_{\mathcal{B}}\mathbf{R}^{T} + \mathbf{R}\mathbf{I}_{\mathcal{B}}\dot{\mathbf{R}}^{T}\right)\omega + \mathbf{I}\dot{\omega} \\ &= \left(\omega^{\times}\mathbf{I} - \mathbf{I}\omega^{\times}\right)\omega + \mathbf{I}\dot{\omega} \\ &= \omega \times \mathbf{I}\omega + \mathbf{I}\dot{\omega}\,,\end{aligned}$$

wobei wir verwendet haben , daß $\mathbf{I} = \mathbf{R}\mathbf{I}_{\mathcal{B}}\mathbf{R}^{T}$ und $\dot{\mathbf{R}} = \omega^{\times}\mathbf{R}$ gilt. Stellt man die Gleichung nach $\dot{\omega}$ um und ersetzt $\mathbf{D}$ entsprechend Gleichung 2.12, so erhält man:

$$\begin{aligned}\dot{\omega} &= \mathbf{I}^{-1}\left(\mathbf{D} - \omega \times \mathbf{I}\omega\right) \\ &= \mathbf{I}^{-1}\left(\sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{F}_i - \omega \times \mathbf{I}\omega\right)\,. \end{aligned} \tag{2.14}$$

2.5.6 Die Bewegungsgleichungen

Die Überlegungen der vorangegangenen Abschnittes versetzen uns in die Lage, die Bewegung eines starren Körpers unter dem Einfluß externer Kräfte zu bestimmen. Dazu verwenden wir die sogenannten *Bewegungsgleichungen* des starren Körpers, die Zeitableitungen der kinematisch relevanten Parameter Position, Orientierung sowie lineare Geschwindigkeit und Winkelgeschwindigkeit beschreiben. Von außerordentlicher Bedeutung sind dabei die dynamische Grundgleichung 2.10, die man auch als *Newton*-Gleichung bezeichnet, sowie die *Euler*-Gleichung aus dem vorangegangenen Abschnitt. Sie beschreiben nämlich die Wirkung von Kräften auf die translatorische Bewegung des Massenschwerpunktes bzw. auf die Rotationsbewegung um diesen. Nimmt man die Gleichungen für die Zeitableitung von Position und Orientierung hinzu, so erhalten wir das folgende Differentialgleichungssystem.

Satz 2.7 (Bewegungsgleichungen des starren Körpers). *Gegeben sei ein starrer Körper, auf den die Kräfte* $\mathbf{F}_1, \ldots, \mathbf{F}_n$ *wirken. Des weiteren bezeichne* $\mathbf{r}_i = \mathbf{x}_i - \mathbf{c}_i$ *die Verschiebung des Angriffspunktes* x_i *der Kraft* $\mathbf{F}_i$ *vom Massenschwerpunkt. Dann läßt sich die Bewegung des*

Körpers durch Integration der folgenden Bewegungsgleichungen bestimmen:

$$(i) \qquad \dot{\mathbf{c}} = \boldsymbol{\nu}\,, \tag{2.15}$$

$$(ii) \qquad \dot{\mathbf{R}} = \boldsymbol{\omega}^{\times}\mathbf{R}\,, \tag{2.16}$$

$$(iii) \qquad \dot{\boldsymbol{\nu}} = \frac{1}{m}\sum_{i=1}^{n}\mathbf{F}_i\,, \tag{2.17}$$

$$(iv) \qquad \dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left(\sum_{i=1}^{n}\mathbf{r}_i \times \mathbf{F}_i - \boldsymbol{\omega}\times\mathbf{I}\boldsymbol{\omega}\right)\,. \tag{2.18}$$

Beweis. Die Aussage folgt unmittelbar aus den Gleichungen 2.3, 2.4, 2.11 und 2.14. □

In der impulsbasierten Dynamiksimulation müssen wir die Bewegung des starren Körpers während der sogenannten *ballistischen Bewegungsphase* beschreiben. Die ballistische Bewegung ist ein Spezialfall der allgemeinen Körperbewegung, da die einzige externe Kraft, der der Körper ausgesetzt ist, die Gravitationskraft ist. Die Bewegungsbahnen des starren Körpers sind in dieser Phase also per Definition *kollisionsfrei.* Einfluß auf die Bewegung nimmt lediglich die „Erdanziehung", die wir als eine auf den Schwerpunkt wirkende Kraft $\mathbf{F} = m\mathbf{g}$ mit Drehmoment $\mathbf{D} = \mathbf{0}$ modellieren können. Der Beschleunigungsvektor $\mathbf{g} = (0, -g, 0)^T$ mit $g = 9,81$ ist in der Nähe der Erdoberfläche nahezu konstant und weist stets in negative y-Richtung, d.h. in Richtung des Erdmittelpunktes. Das Skalar g heißt *Erdbeschleunigung*. Die Beschränkung auf die Gravitationskraft als einzigen externen Einflußfaktor vereinfacht die Bewegungsgleichungen. Gleichung (iii) können wir nun folgendermaßen schreiben:

$$\dot{\boldsymbol{\nu}} = \frac{1}{m}m\mathbf{g} = \mathbf{g} = \text{const}\,.$$

Die Integration dieser Gleichung über das Zeitintervall $[0, t]$ liefert die Gleichung der linearen Geschwindigkeit und der Position des Körpers zum Zeitpunkt t:

$$\boldsymbol{\nu}(t) = \boldsymbol{\nu}(0) + \mathbf{g}t\,, \tag{2.19}$$

$$\mathbf{c}(t) = \mathbf{c}(0) + \boldsymbol{\nu}(0)t + \frac{1}{2}\mathbf{g}t^2\,. \tag{2.20}$$

Da die Gravitationskraft kein Drehmoment induziert vereinfacht sich auch Gleichung (iv):

$$\dot{\boldsymbol{\omega}} = -\mathbf{I}^{-1}\big(\boldsymbol{\omega}\times\mathbf{I}\boldsymbol{\omega}\big)\,.$$

Der Dynamikzustand des Systems kann anhand der Bewegungsgleichungen zu jedem Zeitpunkt einer ballistischen Bewegungsphase ermittelt werden. Die Differentialgleichung zur Bestimmung der aktuellen linearen Geschwindigkeit und Position eines Körpers kann dabei geschlossen gelöst werden. Lediglich die Angabe der Winkelgeschwindigkeit und Orientierung erfordert die numerische Lösung eines gekoppelten Differentialgleichungssystems erster Ordnung. Dazu verwenden wir das in [PTVF92] vorgestellte RUNGE-KUTTA-Verfahren mit variabler Schrittweitenkontrolle.

2.6 Einführung in die Kollisionserkennung

Die Aufgabe der Kollisionserkennung im Rahmen der Dynamiksimulation von Festkörpern besteht darin, die fundamentale Eigenschaft der *Nichtdurchdringbarkeit* der Körper sicherzustellen. Das Kollisionserkennungsproblem ist inhärent geometrischer Natur. Es hat seinen Ursprung im Bereich *Computational Geometry*, wo man sich beispielsweise für die explizite Konstruktion des Schnitts von Polyedern interessiert. Erst durch die Betrachtung des Problems im Rahmen der *Robotik* wurde die Zeitdimension in die Problemformulierung einbezogen. Im Rahmen von Bewegungsplanungsaufgaben genügt es nämlich nicht mehr, statische Schnittanfragen beantworten zu können, sondern hier gewinnen weitergehende Informationen, wie beispielsweise die Bestimmung des Kollisionszeitpunktes zweier Körper an Bedeutung. Anforderungen und Rahmenbedingungen in beiden Gebieten sind jedoch kaum mit denen im Kontext der Dynamiksimulation vergleichbar. Während im Bereich Computational Geometry auf eine *dynamische* Betrachtung der Problemstellung ganz verzichtet wird, stehen in der Robotik offline-Berechnungen im Mittelpunkt. Anforderungen hinsichtlich *Echtzeitfähigkeit in komplexen Szenarien* mit vielen Objekten sowie *Unempfindlichkeit gegenüber schlecht konditionierten Eingabedaten* werden nicht berücksichtigt. Die Erkennung von Kollisionen spielt jedoch in Dynamiksimulationen nicht nur konzeptionell eine zentrale Rolle. Sie stellt im allgemeinen auch den „Flaschenhals“ des Gesamtsystems dar. Bestes Beispiel hierfür sind impulsbasierte Simulationssysteme. Noch anspruchsvoller ist die Situation im Fall von Systemen, die dem Benutzer haptisches Feedback liefern. Diese erfordern aufgrund ihrer Abtastrate mindestens 1000 Kollisionserkennungstests pro Sekunde.

2.6.1 Definition der verschiedenden Problemstellungen

In einer simulierten Welt $\mathcal{W} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$, in der sich Körper $\mathcal{K}_1, \ldots, \mathcal{K}_n$ bewegen, kann es geschehen, daß Trajektorien überlappen, d.h. daß Körper sich schneiden. Den Durchdringungsvorgang müssen wir dabei erkennen, da die Überlappungsfreiheit eine grundlegende Eigenschaft der von uns simulierten Festkörper darstellt. Dies führt zum Problem der Kollisionserkennung. Die Bezeichnung „Kollisionserkennung“ hat sich dabei zu einem Überbegriff für verschiedene Fragestellungen mit ganz unterschiedlichem Anwendungshintergrund entwickelt. Im wesentlichen lassen sich jedoch zwei Problemklassen unterscheiden.

1. *Das statische Kollisionserkennungsproblem*
   Im Rahmen des statischen Kollisionserkennungsproblems werden alle Überlappungen bestimmt, die zwischen Körpern einer vorgegebenen Welt vorliegen.
   Gegeben: Die Welt $\mathcal{W} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$
   Gesucht: Die Menge aller überlappenden Körperpaare:
   $S := \{(\mathcal{K}_i, \mathcal{K}_j) \in \mathcal{W} \times \mathcal{W} \mid \mathcal{K}_i \cap \mathcal{K}_j \neq \emptyset\}$.

2. *Das dynamische Kollisionserkennungsproblem*
   Das dynamische Kollisionserkennungsproblem legt die kontinuierlichen Bewe-

gungen der Körper ausgehend von einer kollisionsfreien Weltkonfiguration zum Zeitpunkt t_0 zugrunde und fragt nach dem frühsten Zeitpunkt $t_{col} > t_0$, zu dem sich die Trajektorien zweier Körper überlappen. Die Antwort besteht aus dem sogenannten Kollisionszeitpunkt t_{col} und der Menge aller überlappender Körperpaare zum Zeitpunkt t_{col}.

Gegeben: Die Konfigurationen der Welt $\mathcal{W} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ zu beliebigen Zeitpunkten $t \geq t_0$, mit $\mathcal{K}_i(t_0) \cap \mathcal{K}_j(t_0) = \emptyset$, $1 \leq i < j \leq n$.

Gesucht: (1) Der Zeitpunkt $t_{col} > t_0$, für den gilt:

(i) $\forall t < t_{col} \forall \mathcal{K}_i, \mathcal{K}_j \in \mathcal{W} : \mathcal{K}_i(t) \cap \mathcal{K}_j(t) = \emptyset$,

(ii) $\mathcal{K}_i(t_{col}) \cap \mathcal{K}_j(t_{col}) \neq \emptyset$.

(2) Die Menge aller überlappenden Körperpaare zum Zeitpunkt $t_{col} : S := \left\{(\mathcal{K}_i, \mathcal{K}_j) \in \mathcal{W} \times \mathcal{W} \,\middle|\, \mathcal{K}_i(t_{col}) \cap \mathcal{K}_j(t_{col} \neq \emptyset\right\}$.

Der *Kollisionszeitpunkt* ist also der frühste Zeitpunkt, zu dem sich zwei Körper der Welt aufgrund einer Bewegung durchdringen. Im Fall einer bloßen Berührung findet keine physikalische Wechselwirkung zwischen den Körpern statt. Alternativ wird der Kollisionszeitpunkt auch als der späteste Zeitpunkt definiert, zu dem die Objekte sich noch nicht durchdringen (vgl. [War99]). Der Unterschied spielt an dieser Stelle keine Rolle und wird deshalb erst im Rahmen der Kollisionsauflösung in Abschnitt 5.2 aufgegriffen. Den von uns beschriebenen Anforderungen im Rahmen der Dynamiksimulation entspricht das dynamische Kollisionserkennungsproblem, wobei jedoch zusätzliche Informationen, wie beispielsweise die in Kontakt stehenden Oberflächenelemente benötigt werden.

Wir wollen uns zunächst mit den theoretischen Ergebnissen aus dem Gebiet der Kollisionserkennung beschäftigen. Diese beziehen sich ausschließlich auf den 2-Polyederfall und verbessern den naiven Algorithmus, der alle $O(n^2)$ Flächenpaare statisch bzw. dynamisch auf Durchdringung testet.

2.6.2 Theoretische Ergebnisse

Die asymptotische Laufzeit $O(n^2)$ des naiven Verfahrens kann nur unter großen algorithmischen Anstrengungen verbessert werden. Wir wollen im folgenden einen Überblick über die wenigen theoretisch relevanten *worst-case* Resultate aus dem Gebiet der Kollisionserkennung geben. Die entsprechenden Algorithmen erzielen subquadratische Laufzeiten, wobei jedoch nur wenige das allgemeine dynamische Kollisionserkennungsproblem lösen. Um die Resultate vergleichbar zu machen, müssen die Einschränkungen, die den Verfahren zugrunde liegen, herausgestellt werden. Diese betreffen zum einen das Körpermodell und zum anderen die Eignung im Rahmen der dynamischen Fragestellung. So existieren beispielsweise Verfahren, deren Eingabe auf konvexe Polyeder beschränkt ist oder deren algorithmische Komplexität lediglich für translatorische Bewegungen der Körper Gültigkeit besitzt (weil beispielsweise eine Vorverarbeitung erforderlich ist).

DOBKIN und KIRKPATRICK zeigen in [DK83], daß mit Hilfe einer hierarchischen Repräsentation der Kollisionstest zweier konvexer Polyeder in Zeit $O(\log^2 n)$ durchgeführt werden kann, wobei n die Zahl der Eckpunkte angibt. Erlaubt man, daß

eines der beiden Polyeder nichtkonvex ist, so löst der Algorithmus von DOBKIN ET AL. [DHKS93] das Erkennungsproblem in Zeit $O(n \log n)$. MEHLHORN und SIMON sowie DOBRINDT ET AL. zeigen in [MS85] bzw. [DMY93], daß diese Komplexitätsschranke selbst bei expliziter Konstruktion des Schnittpolyeders eingehalten werden kann. Im Fall sogenannter c-*isoorientierter Polyeder*, deren Kantenrichtungen aus einer Menge von c festen Richtungen stammen, kann der Kollisionstest in Zeit $O(c^2 n \log n)$ durchgeführt werden [Sch94]. Allen Resultaten ist gemeinsam, daß die worst-case Laufzeit der zugrundeliegenden Verfahren lediglich unter translatorischen Bewegungen ihre Gültigkeit behält. Keiner dieser Algorithmen kann also für den allgemeinen Fall nichtkonvexer Polyeder, die sowohl translatorischen als auch rotatorischen Bewegungen unterliegen, eine subquadratische Laufzeit nachweisen. Diese bis dato offene Frage, ob der naive Algorithmus in einer worst-case Betrachtung verbessert werden kann, wurde von SCHÖMER und THIEL in [ST96] beantwortet. Sie zeigen, daß mit Hilfe von parametrischer Suche und geeigneten Multi-Level-Datenstrukturen eine asymptotische Laufzeit von $O(m^{\frac{8}{5}+\varepsilon})$ für die translatorische bzw. $O(m^{\frac{5}{3}+\varepsilon})$ für die rotatorische Bewegung erzielt werden kann. Die Autoren beweisen weiter, daß selbst für eine polynomielle Trajektorie eine subquadratische Laufzeit $o(n^2)$ möglich ist.

2.6.3 Klassifikation praxisrelevanter Ansätze

Aufgrund der großen Zahl von Algorithmen, die dem Oberbegriff Kollisionserkennung zuzuordnen sind, wollen wir die bis dato entwickelten, praktisch orientierten Verfahren systematisieren.

Einteilung nach der gelösten Problemstellung

Die beiden zentralen Problemstellungen, die unter den Begriff Kollisionserkennung fallen, wurden in Abschnitt 2.6.1 vorgestellt. HELD, KLOSOWSKI und MITCHELL [HKM95] unterteilen die Kollisionserkennungsverfahren entsprechend der von ihnen gelösten Aufgabenstellung in drei Kategorien:

1. *Die statische Kollisionserkennung*
   Die Verfahren der statischen Kollisionserkennung lösen das statische Kollisionserkennungsproblem, indem sie für jedes Körperpaar einer vorgegebenen Weltkonfiguration die Kollisionsfrage entscheiden.

2. *Die pseudodynamische Kollisionserkennung*
   Diese Verfahren benutzen statische Kollisionserkennungsalgorithmen, um zu *diskreten Zeitpunkten* die Bewegung der Körper auf Kollision zu testen. Der Nachteil dieser Verfahren besteht darin, daß Kollisionen häufig übersehen werden, selbst wenn die Zeitintervalle zwischen den statischen Kollisionstests klein gewählt werden. Das klassische Beispiel ist das der Kugel, die auf eine Wand geschossen wird. Egal wie fein die Kollisionserkennung das relevante Zeitintervall auflöst, die Geschwindigkeit und die Dicke der Wand kann so gewählt werden, daß

die Kollision nicht erkannt wird. Da diese Algorithmen das dynamische Kollisionserkennungsproblem nur unzureichend lösen, spricht man von sogenannten pseudodynamischen Kollisionserkennungsverfahren.

3. *Die dynamische Kollisionserkennung*
   Die dynamischen Kollisionserkennungsalgorithmen überprüfen die Kollisionsfreiheit der Körper nicht zu diskreten Zeitpunkten ihrer Bewegung. Stattdessen betrachten sie das durch die Bewegung *überstrichene Volumen* eines Körpers und testen dieses auf Kollision mit den überstrichenen Volumina anderer Körper. Das Volumen muß dazu nicht zwangsläufig explizit konstruiert werden, da es implizit durch eine Approximation der Körpertrajektorien angenähert werden kann. Diese Vorgehensweise erlaubt, größere Simulationsschrittweiten zu wählen, ohne das Risiko einzugehen, Kollisionen zu übersehen.

Da das Gebiet der Dynamiksimulation starrer Körper erst in den letzten Jahren Einfluß auf das Design von Kollisionserkennungsalgorithmen genommen hat, sind nahezu alle Verfahren statischer bzw. pseudodynamischer Natur. Die anspruchsvollen Anforderungen der dynamischen Kollisionserkennung erfüllen nur wenige Ansätze, die wir daher explizit erwähnen wollen. Ein erster Ansatz ist die Konstruktion des *vierdimensionalen Hyperpolyeders* von CANNY [Can94], der die Kollisionserkennung auf den in der Zeit überstrichenen Polyedervolumina durchführt. Eine ähnliche Idee stellen die *space-time bounds* von HUBBARD [Hub95] dar, die Informationen über die Geschwindigkeit und Beschleunigung eines Körpers nutzen, um ein konservatives Volumen zu definieren, in dem sich der Körper innerhalb des Zeitintervalls befindet. Dieses Volumen wird in einen vierdimensionalen Polyeder, das sogenannte *Hypertrapezoid* eingeschlossen, dessen Kollisionen mit anderen Hypertrapezoiden untere Schranken für den Kollisionszeitpunkt definieren. Letztere werden zur Steuerung einer statischen Kollisionserkennung verwendet und erlauben eine bis auf ein Δ genaue Approximation des Kollisionszeitpunktes. Den Wert Δ bezeichnet HUBBARD als *minimale zeitliche Auflösung* der Kollisionserkennung.

Die Idee der *unteren Kollisionszeitschranken* ist eine beliebte Technik, wenn es darum geht, eine dynamische Kollisionserkennung zu realisieren [CK86, Lin93, Hub95]. Eine solche Schranke für zwei Körper definiert einen frühsten Kollisionszeitpunkt und somit einen Zeitraum, in dem die beiden Objekte nicht miteinander kollidieren können. Daher kann die Bewegung der Körper in diesem Zeitintervall simuliert werden, ohne daß man kontaktinduzierte Bewegungseinflüsse beachten muß. Ist der frühste Kollisionszeitpunkt erreicht, ist die Garantie der berechneten Kollisionszeitschranke nicht mehr gültig und ein Kollisionstest ist unumgänglich. Falls dabei keine Überlappung der beiden Körper festgestellt wird, kann eine präzisere Schranke aufgrund aktuellerer geometrischer und dynamischer Informationen bestimmt werden. Diese Technik kann mit Hilfe einer Prioritätswarteschlange auf das Problem der *n-Körper-Kollisionserkennung* verallgemeinert werden. Die Bestimmung der unteren Kollisionszeitschranken setzt voraus, daß Maximalwerte für die Geschwindigkeiten und Beschleunigungen der Körper verfügbar und geometrische Informationen über den aktuellen Abstand der Objekte bereits berechnet sind. Das Problem der Vorgehensweise besteht gerade in dieser

Notwendigkeit, Dynamikinformationen des nächsten Zeitschritts zu kennen, bevor das Intervall tatsächlich simuliert wird. Dies macht die Vorgehensweise für eine interaktive Simulation, in der der Benutzer in das dynamische Geschehen eingreifen kann, ungeeignet. Der entscheidende Vorteil, den diese Technik gegenüber allen anderen Verfahren bietet, ist die *absolute Garantie, daß jede Kollision im zugrundeliegenden Zeitintervall erkannt wird*. Alle anderen Verfahren können diese Garantie nur unter bestimmten plausiblen Bewegungsannahmen geben. Diese Betrachtung dient gleichzeitig als Motivation für das in Kapitel 3 untersuchte *Abstandsberechnungsproblem* zweier Polyeder. Im Kontext einer Dynamiksimulation kann also ein statisches Abstandsberechnungsverfahren zur Lösung des dynamischen Kollisionserkennungsproblems eingesetzt werden (vgl. Kapitel 4).

Die Suche nach dem frühsten Kollisionszeitpunkt t_{col} zweier Körper entspricht dem Problem der *Nullstellensuche auf der Distanzfunktion*. Mit Hilfe unterer Kollisionszeitschranken kann t_{col} ausgehend von einer kollisionsfreien Konfiguration der Körper derart angenähert werden, daß der gesuchte Zeitpunkt niemals überschritten wird. Abstandsberechnungsverfahren, die neben der Distanz im kollisionsfreien Fall ein Maß für die *Durchdringungstiefe* berechnen können [Cam97, Mir97b], erlauben zudem den Einsatz der klassischen Verfahren zur Nullstellensuche. Falls die Objekte sehr nahe sind, arbeiten solche Verfahren deutlich effizienter als die Algorithmen, die eine einseitige Approximation des Kollisionszeitpunktes mittels Kollisionszeitschranken durchführen.

Eine ganz andere Vorgehensweise liegt dem Verfahren von ECKSTEIN und SCHÖMER [Eck98, ES99] zugrunde, welches die in Abschnitt 2.6.4 vorgestellte Idee der *Hüllkörperhierarchie* benutzt, um einen dynamischen Kollisionstest durchzuführen. Dabei werden die bis dato ausschließlich statisch verwendeten Ausschlußtests auf der Basis von Hüllkörpern, sowie die elementaren Punkt-Fläche und Kante-Kante-Kollisionstests auf Zeitintervalle verallgemeinert. Diesen Tests werden gewisse *Bewegungsannahmen* zugrundegelegt, die zum einen eine einfache Berechnung der von den Hüllkörpern überstrichenen Volumina und zum anderen eine effiziente Nullstellensuche auf der Distanzfunktion ermöglichen.

Einteilung nach den Kollisionserkennungsphasen

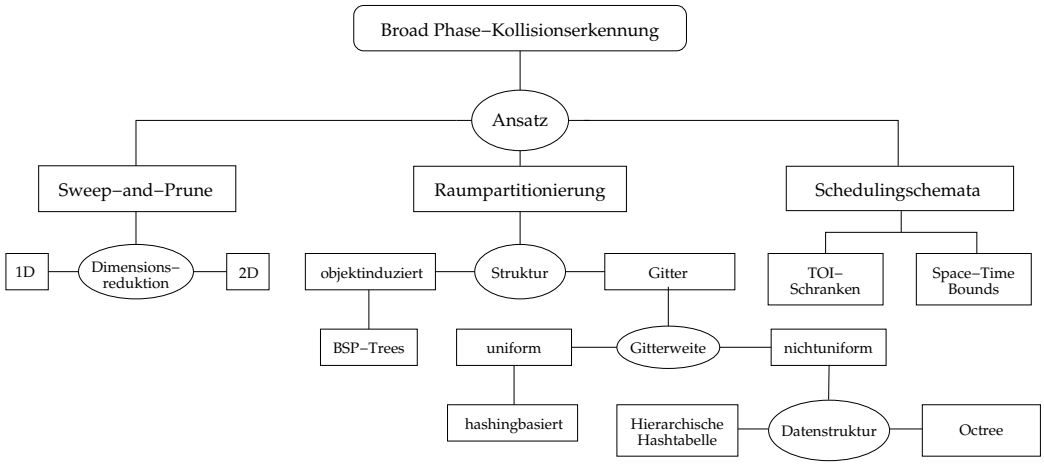
Die Komplexität des Kollisionserkennungsproblems wird letztendlich durch zwei Problemparameter erfaßt:

- Die *Anzahl der Objekte* in der Welt: m.
- Die *Komplexität der Körperbeschreibung* (z.B. Zahl der Oberflächenelemente: n).

Diese beiden Größen führen zu zwei unabhängigen Problemkreisen im Rahmen des Kollisionserkennungsproblems (vgl. [Hub95]):

1. Reduktion der $O(m^2)$ paarweisen Kollisionstests (*all-pairs weakness*).
2. Reduktion des Einflusses der Beschreibungskomplexität auf die Laufzeit eines einzelnen Tests gegenüber dem naiven $O(n^2)$ Verfahren (*pair-processing weakness*).

Abbildung 2.9 Klassifikation der Ansätze zur broad phase-Kollisionserkennung.



Die meisten praktisch relevanten Kollisionserkennungssysteme unterscheiden daher zwei Phasen der Kollisionserkennung, die den beiden angesprochenen Teilproblemen gewidmet sind. Es haben sich dabei die Bezeichnungen von HUBBARD [Hub95] durchgesetzt, der im Rahmen seines space-time bound Ansatzes zwischen der sogenannten *broad phase* und der *narrow phase* differenziert. In der broad phase werden solche Objektpaare ausgeschlossen, die so weit voneinander entfernt sind, daß bereits einfachste hinreichende Tests eine Kollisionsfreiheit zu einem festen Zeitpunkt oder innerhalb eines vorgegebenen Zeitraums garantieren. In der sich anschließenden narrow phase werden die verbleibenden Objektpaare mit Hilfe eines exakten Verfahrens auf Kollision hin überprüft.

2.6.4 Bisherige Arbeiten

Wir wollen im Rahmen dieser Übersicht die Klassifikation bezüglich der Kollisionserkennungsphasen zugrunde legen und dabei ausschließlich statische Verfahren vorstellen. Die dynamischen Verfahren bzw. generelle Techniken zu einer diesbezüglichen Erweiterung statischer Abstandsberechnungsverfahren haben wir bereits in 2.6.3 dargestellt.

Die broad phase-Kollisionserkennung

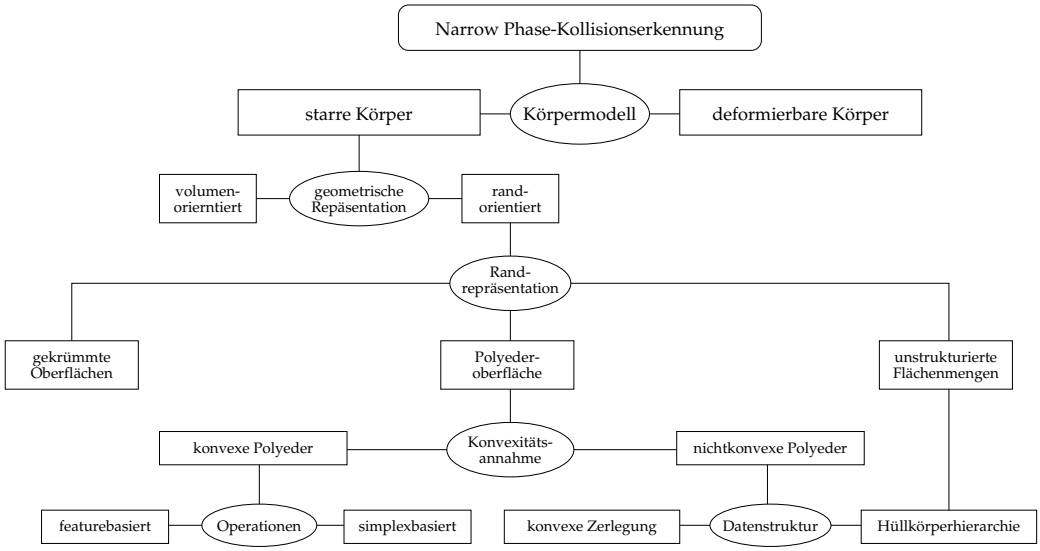
LIN schlägt in [Lin93] vor, die Objekte in achsenorientierte Quader einzuhüllen. Die Projektion der Quader auf die Koordinatenachsen (*1D-Sweep-and-Prune*) bzw. Koordinatenebenen (*2D-Sweep-and-Prune*) erlaubt, alle Paare sich schneidender Boxen anhand von Überlappungen in niedrigeren Dimensionen effizient zu bestimmen. Bei den Körpern, deren umhüllende Quader sich schneiden, handelt es sich um potentielle Kollisi-

onspaare für die zweite Phase der Kollisionserkennung.

Eine ähnliche Idee liegt den sogenannten *Raumpartitionierungsverfahren* zugrunde. Dabei wird der Raum, den das Simulationsszenario einnimmt, in einzelne Regionen unterteilt. Objektpaare, die sich in verschiedenen Regionen befinden, können als Kollisionskandidaten vorzeitig ausgeschlossen werden, während alle anderen Paare in der narrow phase untersucht werden müssen. Die Aufteilung des Raumes kann sich dabei an den Objekten der Szene orientieren, oder völlig unabhängig von deren räumlicher Verteilung erfolgen. Ein Beispiel für eine solche *objektinduzierte* Unterteilung ist der sogenannte *Binary Space Partitioning Tree* (*BSP-tree*) von FUCHS ET AL. [FKN80] zur Lösung des *hidden-surface-removal*-Problems. Dabei handelt es sich um einen binären Baum, der in jedem Knoten einen durch Halbebenen definierten Raum unter Hinzunahme einer neuen trennenden Ebene in zwei Teilräume unterteilt. Die Trennebene ist dabei die unterstützende Ebene eines Polygons aus der Menge aller Objektpolygone, die sich in dem zu unterteilenden Raum befinden. Unabhängig von Lage und Gestalt der Objekte zerlegen die *gitterbasierten* Verfahren den Raum in Würfel (Voxel) gleicher (*uniforme Raumpartitionierung*) bzw. unterschiedlicher Seitenlänge (*nichtuniforme Raumpartitionierung*). Im Fall der uniformen Unterteilung werden die Objekte mittels *Hashing* den von ihnen besetzten Raumwürfeln zugeteilt [Ove92, ZOMP93, Mir96b]. Dies erlaubt es, die Objektpaare, die in gleichen Voxeln liegen, effizient als potentielle Kollisionspaare zu identifizieren. Eine nichtuniforme Raumpartitionierung kann auf analoge Weise durch eine *hierarchische Hashtabelle* realisiert werden. „Hierarchisch“ bedeutet in diesem Zusammenhang, daß mehrere Hashtabellen zur Verfügung stehen, die Raumaufteilungen unterschiedlicher Gitterweite repräsentieren [Ove92, Mir96b]. Somit kann die Auflösung der Raumpartitionierung besser an unterschiedliche Objektgrößen angepaßt und die Kosten der narrow phase-Tests zu der Distanz der Körper in Beziehung gesetzt werden. Eine andere Datenstruktur für Raumpartitionierungen unterschiedlicher Gitterweiten stellen die sogenannten Octrees [Hah88, MW88, Zac94] dar. Dabei handelt es sich um eine Hierarchie achsenorientierter Quader, wobei auf jeder Stufe der Hierarchie der gesamte Raum partitioniert wird, der Objekte enthält. Auf der obersten Stufe wird der relevante Raum in einen einzigen Quader eingeschlossen und anschließend in acht Quader zerlegt, indem jede Kante des übergeordneten Quaders unterteilt wird. Durch Wiederholung dieses Unterteilungsprozesses erhält man immer feinere Raumpartitionierungen auf den einzelnen Stufen der Hierarchie.

Eine ganz andere Klasse von Verfahren verwendet die in 2.6.3 vorgestellte Kollisionszeitschrankentechnik. Die potentiellen Kollisionszeitpunkte aller Körperpaare werden mittels eines *Schedulingschemas* [CK86, Lin93] auf der Zeitachse angeordnet, so daß stets nur das Objektpaar auf Kollision getestet werden muß, das den frühsten möglichen Kollisionszeitpunkt definiert. In Hinblick auf die Berechnung der Kollisionszeitschranken lassen sich das *space-time bound-Verfahren* von HUBBARD [Hub95] und dessen Weiterentwicklung durch SAUER [Sau95], sowie die *Kollisionsheap*-Ansätze von VON HERZEN ET AL. [BBZ90], SNYDER ET AL. [SWF+93] und MIRTICH [Mir96b] unterscheiden.

Abbildung 2.10 Klassifikation der Ansätze zur narrow phase-Kollisionserkennung.



Die narrow phase-Kollisionserkenung

Starre und deformierbare Körper

Die drastischste Einschränkung, die den meisten Kollisionserkennungsverfahren zugrunde liegt, ist die Idealisierung der realen Objekte als *starre Körper*. Ähnlich wie im Fall der Kollisionsbehandlung ist diese Annahme auch im Rahmen der Kollisionserkennung unumgänglich, will man Echtzeitanforderungen genügen. Kollisionserkennung für *deformierbare Körper* besitzt jedoch zahlreiche Anwendungsgebiete. Man denke beispielsweise an die Simulation der Verlegung von Kabelbäumen in Automobilen oder an Kleidungssimulationen im Animationsbereich. Einen Überblick über den Stand des Forschungsgebiets liefern die Arbeiten aus [MW88, SWF+93, Gas93, SKTK95, KSTK98, HDLM96].

Polygonale und gekrümmte Oberflächen

Die geometrische Repräsentation der Körper wird im allgemeinen durch die globale Anwendung bestimmt. Während *volumenorientierte Repräsentationen* beispielsweise in den aufkommenden medizinischen Anwendungsgebieten bevorzugt werden, wurden Kollisionserkennungsverfahren bisher primär für *randorientierte Körpermodelle* entwickelt. Dabei unterscheidet man zwischen einer großen Klasse von Algorithmen für *polygonale Oberflächen* und einer eher kleinen Gruppe von Verfahren für *gekrümmte Oberflächen*. Dies überrascht eigentlich, da die Eingabedaten im allgemeinen aus CAD-Systemen stammen und diese bekanntlich auf Körpern mit gekrümmten Oberflächen operieren. Bei der Konvertierung in polygonale Randdarstellungen gehen oft wichtige

topologische Eigenschaften wie der Zusammenhang der Oberflächenelemente verloren. Hinzu kommt, daß die Beschreibungskomplexität der Modelle durch die Approximati-on gekrümmter Flächen drastisch ansteigt. Der Grund für die große Beliebtheit polygonaler Randdarstellungen, liegt in der geometrischen Einfachheit der Oberflächenelemente. Diese Eigenschaft erlaubt die Entwicklung einfacher, effizienter Algorithmen unter Rückgriff auf eine breite theoretische Basis und einen großen praktischen Erfahrungsschatz aus dem Bereich Computational Geometry. Für gekrümmte Oberflächen werden dagegen im allgemeinen komplizierte, numerische Verfahren benötigt, die nicht die Effizienz der Verfahren auf polygonalen Oberflächen erzielen. Ein weiterer Grund, der bisher für den durchgehenden Einsatz polygonaler Modelle sprach, ist die gute hardwareseitige Unterstützung der Visualisierung polygonaler Strukturen. Entwicklungen auf dem Gebiet des hardwarebeschleunigten Renderings von NURBS-Oberflächen und die zunehmende Bedeutung von *Virtual Prototyping*, das einen intensiven Austausch zwischen CAD- und Simulationssystem erfordert, machen den Einsatz von Kollisionserkennungsalgorithmen für gekrümmte Oberflächen zunehmend interessanter. Bisherige Ansätze von SNYDER ET AL., VON HERZEN ET AL., BARAFF [Bar90] sowie LIN und MANOCHA werden in [BBZ90], [SWF+93] bzw. [LM93] beschrieben.

Polyederoberflächen und unstrukturierte Flächenmengen

Wie wir bereits erwähnt haben, erzwingt die Verwendung polygonal begrenzter Körpermodelle die Konvertierung der CAD-Quelldaten in die erforderliche Körperrepräsentation. Das Ergebnis einer solchen Konvertierung ist im allgemeinen kein Polyeder, sondern vielmehr eine *Ansammlung planarer Flächen*. Während der Verlust von Abschluß- und Zusammenhangseigenschaft für die Visualisierung keine große Einschränkung bedeutet, scheiden Kollisionserkennungsverfahren, die *Inzidenzinformationen* der Flächen ausnutzen, unter diesen Randbedingungen aus. Solange die Probleme im Zusammenhang mit der Datenkonvertierung nicht gelöst sind, stellt der Verzicht auf die Verwendung von Inzidenzinformationen ein Praktikabilitätskriterium für Kollisionserkennungssysteme dar. Dennoch beruht die Vorgehensweise vieler Ansätze, insbesondere im Fall konvexer Polyeder, auf der Unversehrtheit der Polyederoberfläche.

Feature- und simplexbasierte Verfahren für konvexe Polyeder

Für die Klasse der *konvexen Polyeder* existieren die effizientesten Kollisionserkennungsverfahren. Konvexität ist eine geometrische Einschränkung, die im Rahmen der Kollisionserkennung und speziell der Abstandsberechnung starrer Körper großen Einfluß auf die Laufzeit der Verfahren besitzt. Mirtich [Mir97a] unterscheidet in diesem Zusammenhang zwischen *feature-* und *simplexbasierten* Verfahren. Die featurebasierten Algorithmen operieren auf den Oberflächenelementen (*features*), d.h. den Eckpunkten, Kanten und Flächen des konvexen Polyeders, während simplexbasierte Verfahren die Simplices der Eckpunktmenge, also die Punkte, Geradensegmente, Dreiecke und Tetraeder, betrachten.

BARAFF stellt in [Bar92] den sogenannten *witness-plane-Algorithmus* vor. Bei der witness-plane zweier disjunkter, konvexer Polyeder handelt es sich um eine *trennende Ebene* der beiden Körper, d.h. um einen Zeugen der Kollisionsfreiheit. Diese Ebene

ist dabei entweder parallel zu einer Fläche der beiden Polyeder, oder zu einem Kantenpaar, das von jeweils einer Kante der beiden Polyeder gestellt wird. Ein featurebasiertes Abstandsberechnungverfahren, welches die Basis der Kollisionsbibliothek *I-Collide* [CLMP95] bildet, stellen LIN und CANNY in [LC91, Lin93] vor. Das Verfahren sucht ausgehend von einem Paar von Oberflächenelementen der beiden Körper die Oberfläche nach demjenigen mit minimalem Abstand ab. Dazu betrachtet der Algorithmus die *externen Voronoiregionen* der Oberflächenelemente. Mit Hilfe der Voronoiregionen kann ein hinreichendes Kriterium für die globale Abstandsminimalität eines Paares von Oberflächenelementen formuliert werden. Ist dieses Kriterium nicht erfüllt, wird das Oberflächenelement, das die Bedingung verletzt, durch ein inzidentes Element ausgetauscht. Dieser Vorgang verringert den bisher gefundenden Polyederabstand. Aufgrund dieser Vorgehensweise wird das Lin-Canny-Verfahren auch als *closest-features tracking-Algorithmus* bezeichnet. Ein schwerwiegender Nachteil des Verfahrens ist das zyklische Verhalten in degenerierten Situationen, und vor allem im Durchdringungsfall. Der *V-Clip-Algorithmus* von MIRTICH [Mir97b] beseitigt diese Einschränkung durch geeignete Modifikationen, die den Algorithmus zudem numerisch stabiler und einfacher machen. Des weiteren liefert das Verfahren ein Maß für die *Durchdringungstiefe* im Kollisionsfall.

Der zentrale simplexbasierte Algorithmus ist der *GJK-Algorithmus*, der von GILBERT, JOHNSON und KEERTHI in [GJK88] beschrieben wird. Dieser bestimmt implizit das *Minkowski-Differenz-Polyeder* der beiden konvexen Ausgangspolyeder und sucht in diesem einen Simplex, der den Ursprung enthält bzw. zu diesem am nächsten liegt. Falls ein Simplex existiert, der den Ursprung enthält, so liegt eine Kollision der Ausgangspolyeder vor, andernfalls liefert der Abstand des nahesten Simplex zum Ursprung die Distanz der beiden Polyeder. Auch dieser Algorithmus ist in der Lage, ein Maß für die *Durchdringungstiefe* der Polyeder anzugeben. RABBITZ [Rab94] und CAMERON [Cam97] haben schließlich das Verfahren hinsichtlich der Ausnutzung temporärer Kohärenz (s.u.) modifiziert und auf diese Weise die Laufzeit in der Praxis verbessert. CHUNG TAT LEUNG [Leu96] ersetzt den Algorithmus von LIN in der I-Collide-Bibliothek durch den Algorithmus von RABBITZ. Die so entstandene Bibliothek ist unter dem *Q-Collide* verfügbar.

Konvexe Zerlegungen und Hüllkörperhierarchien für nichtkonvexe Polyeder

Aufgrund der Effizienz der Verfahren für konvexe Polyeder verwenden einige Kollisionserkennungssysteme *konvexe Zerlegungen* ihrer Polyedermodelle [Lin93, Leu96, Cam97, Mir97b]. Unter der Annahme, daß die beiden Polyeder in r bzw. s konvexe Teile zerlegt werden können [SN86, CP89, BD92], läßt sich der Kollisionstest zwischen den Ausgangspolyedern auf $r \cdot s$ Tests zwischen den konvexen Teilkörpern zurückführen. Unglücklicherweise ist das Problem, einen gegebenen Polyeder in eine minimale Zahl von konvexen Teilpolyedern zu zerlegen, NP-hart [Lin82], wobei in [Cha84] gezeigt wird, daß ein solcher Polyeder mit n Knoten stets in $O(n^2)$ konvexe Teile unterteilt werden kann. Betrachtet man nun den wichtigen Spezialfall der Zerlegung in Tetraeder, so ist die Zahl der Tetraeder im worst-case quadratisch in der Zahl der Kanten, die eine lokale Konvexitätsverletzung anzeigen (Winkel zweier inzidenter Flächen ist

auf der Außenseite kleiner 180°) [Cha84]. Es existiert dabei ein asymptotisch optimaler Algorithmus für Polyeder mit Genus null [CP89]. Eine Implementierung eines solchen Verfahrens erweist sich jedoch als äußerst diffizil. Es ist derzeit keine praktisch verwendbare Implementierung bekannt. Der Versuch, den Lin-Canny-Algorithmus auf nichtkonvexe Polyeder ohne den Umweg einer konvexen Zerlegung zu übertragen [PML95], scheint aufgrund zahlreicher degenerierter Fälle und unzureichender numerischer Stabilität gescheitert zu sein.

Eine echte Alternative zu den Verfahren für konvexe Polyeder stellen die auf *Hüllkörperhierarchien* beruhenden Kollisionserkennungsalgorithmen dar. Hüllkörperhierarchien sind Bäume, deren Knoten mit einer Flächenmenge sowie einem einfachen *geometrischen Primitiv* assoziiert sind. Dieses Primitiv überdeckt die Flächenmenge des Knotens und wird daher als *Hüllkörper* bezeichnet. Die Wurzel des Baumes repräsentiert die vollständige Flächenmenge des Körpers, die zu den Blättern hin zunehmend feiner unterteilt wird. Dabei bilden die Kinder eines Knotens eine Partition der Flächenmenge des Vaterknotens. Der Hüllkörper des Vaterknotens wird durch zwei neue Hüllkörper auf der Ebene der Kinderknoten ersetzt. Auf diese Weise nimmt die Approximationsgüte der Oberflächenumhüllung von Hierarchiestufe zu Hierarchiestufe zu, wobei die Blattknoten die feinste Überdeckung der Körperoberfläche induzieren. Die Verfahren unterscheiden sich im wesentlichen hinsichtlich der geometrischen Gestalt der Hüllkörpertypen, dem Verfahren zum Aufbau der Hierarchie, sowie der Durchmusterungsstrategie des Baumes bei der Beantwortung der Überlappungsfrage. Als Hüllkörpertypen werden *Kugeln* [Hub95, PG95, Qui94], *achsenorientierte Boxen* [ZF95, Zac97, Zac98], *beliebig orientierte Boxen* [GLM96], *Fixed Directions Hulls* [HKM+96, KZ97, Zac98, Kon98] und *Kugelkappen* [KPLM98] eingesetzt. Die Hüllkörperhierarchien erlauben eine einfache *Heuristik* zur Beschleunigung des naiven Kollisionstests, welcher alle Flächenpaare auf Überlappung prüft. Statt die gesamte Flächenmenge oder Teile von dieser zu testen, werden zunächst die entsprechenden Hüllkörper auf Kollision geprüft. Überlappen diese nicht, so kann eine Kollision zwischen den von ihnen eingehüllten Flächenmengen ausgeschlossen werden. Andernfalls kann eine feinere Überdeckung der Flächenmengen auf der nächsten Ebene der Hüllkörperhierarchie betrachtet werden.

Temporäre, geometrische und lokale Kohärenz

In Dynamiksimulationen löst man bestimmte Fragestellungen, wie sie beispielsweise von dem Kollisionserkennungsproblem aufgeworfen werden, in aufeinanderfolgenden Zeitschritten immer wieder neu. Die entsprechenden Probleminstanzen sind oft verwandt, weil die Schrittweiten im allgemeinen so gewählt werden, daß die Objekte sich nur geringfügig in dem entsprechenden Zeitintervall bewegen. Man spricht in diesem Fall von *temporärer Kohärenz* zwischen den Probleminstanzen. Es macht daher Sinn, Resultate des vorangegangenen Zeitschritts zur Beschleunigung der aktuellen Problemlösung zu nutzen, d.h. eine zeitliche *Kostenamortisierung* zu erlauben. Sind die Fragen primär geometrischer Natur wie im Fall der Kollisionserkennung, so impliziert temporäre Kohärenz auch *geometrische Kohärenz*. Liegt nämlich temporäre Kohärenz vor, so befinden sich beispielsweise die aktuellen nahesten Oberflächenelemente zweier Polyeder in der Nähe der Elemente des letzten Zeitschritts. Diese Annahme machen sich der

closest-features tracking-Algorithmus [Lin93] sowie der V-Clip-Algorithmus [Mir97b] zu Nutze. Sie verwenden die zwischengespeicherten Oberflächenelemente des letzten Zeitschritts als Startwerte des Algorithmus' und können ausgehend von diesen die aktuellen nahesten Elemente in nahezu konstanter Zeit finden. Das gleiche Prinzip liegt dem witness-plane-Algorithmus von BARAFF [Bar92] sowie dem Q-Collide Verfahren von CHUNG TAT LEUNG [Leu96] zugrunde. Diese versuchen, die Kollisonsfreiheit zweier konvexer Polyeder ausgehend von der trennenden Ebene des letzten Zeitschritts zu verifizieren. Falls dies nicht sofort gelingt, erlauben wenige lokale Anpassungen eine neue Trennebene für die aktuelle Lage der Polyeder zu finden. Auch der GJK-Algorithmus gestattet in der Variante von CAMERON [Cam97], temporäre Kohärenz durch Caching-Strategien auszunutzen. Seine empirische Laufzeit ist aufgrund dieser Modifikationen ebenfalls nahezu unabhängig von der Beschreibungskomplexität der Polyeder. Räumliche Kohärenz drückt die Annahme aus, daß die Objekte im allgemeinen stark gestreut im Raum verteilt liegen. Viele Raumregionen sind daher von nur einem oder gar keinem Objekt besetzt. Diese Eigenschaft machen sich gitterorientierte Raumpartitionierungsverfahren zu Nutze. Bestes Beispiel sind die hashingbasierten Verfahren aus [Ove92, ZOMP93, Mir96b].

2.6.5 Anforderungen an die Kollisionserkennung

Kollisionserkennung ist eine der Schlüsselkomponenten des Dynamiksimulationssystems. Die *verläßliche Erkennung von Kollisionen* ist notwendige Voraussetzung für die Erzeugung physikalisch korrekter Simulationsdaten. Reale Kollisionen, die in der Simulation nicht aufgelöst wurden, verfälschen die Simulationsergebnisse nachhaltig. Mangelnde Zuverlässigkeit hat oftmals als Ursache numerische Instabilität. Numerische Ungenauigkeiten resultieren dabei aus schlecht konditionierten, geometrischen Eingabedaten, Rundungsfehlern oder numerischen Schwächen der Algorithmen in degenerierten Kontaktsituationen. Solche entarteten Kontaktsituationen sind aber gerade in den von uns skizzierten Anwendungsgebieten der Dynamiksimulation, wie beispielsweise beim Virtual Prototyping oder der Montageplanung, nicht selten. Daher muß der *numerischen Robustheit* der Verfahren besondere Aufmerksamkeit geschenkt werden.

Pseudodynamische Verfahren der Kollisionserkennung scheiden im Umfeld der Echtzeitsimulation aus, da sie bei kleiner Schrittweite zwar relativ zuverlässig arbeiten, doch den Anforderungen hinsichtlich der Rechenzeit nicht genügen können. Somit kommt lediglich ein *dynamisches Kollisionserkennungskonzept* in Frage, das größere Schrittweiten bei zuverlässiger Erkennungsrate erlaubt.

Effizienz ist die zweite zentrale Anforderung, der man bei der Entwicklung von Kollisionserkennungsalgorithmen Beachtung schenken muß. Die Kollisionserkennung ist traditionell der Flaschenhals der Dynamiksimulation. Insbesondere in Verbindung mit der Ansteuerung von Force-Feedback-Systemen, zu deren sinnvollem Einsatz eine kontinuierliche Rate von mindestens 1000 Kollisionserkennungstests pro Sekunde erforderlich ist, wird die Bedeutung dieser Effizienzforderung offensichtlich. Wir wollen unsere Ziele in dieser Arbeit nicht ganz so hoch stecken und eine für die Echtzeitvisua-

liserung erforderliche Rate von 20-25 Tests pro Sekunde realisieren.

Neben einer effizienten Gestaltung des eigentlichen Kollisionserkennungsverfahrens sollten die im dynamischen Umfeld zu erwartenden Beschleunigungsmöglichkeiten aufgrund *temporärer Kohärenz* genutzt werden.

In echtzeitfähigen Systemen stellt die *Skalierbarkeit* der Algorithmen eine nützliche Eigenschaft dar. Die Verfahren sollten in der Lage sein, ihren Zeitverbrauch entsprechend des zur Verfügung stehenden Zeitbudgets einzuteilen.

Eine rein anwendungsorientierte Anforderung ist die Eignung für eine möglichst *große Klasse von Körperrepäsentationen*. Das Kollisionserkennungsverfahren sollte aus Gründen der Praktikabilität zumindest allgemeine Polyeder handhaben können. Wünschenswert wäre darüber hinaus ein Verzicht auf die Verwendung von Flächeninzidenzinformationen, um auf den in der Praxis häufig auftretenden unstrukturierten Flächenmengen operieren zu können. Eine weitergehende Forderung wäre schließlich die einfache Erweiterbarkeit auf Körper mit gekrümmten Oberflächen.

Wir können somit die folgenden Anforderungen an das zu entwickelnde Kollisionserkennungsystem festhalten:

- Zuverlässigkeit,
- numerische Robustheit,
- Echtzeitfähigkeit,
- Ausnutzung temporärer Kohärenz,
- Skalierbarkeit,
- Eignung für möglichst allgemeine Körperrepräsentation.

Ein dynamischer Kollisionserkennungsansatz auf der Basis statischer Abstandsberechnung

3 Statische Abstandsberechnung starrer Körper

In diesem Teil der Arbeit wollen wir ein dynamisches Kollisionserkennungsverfahren für Körper mit polygonaler Randbeschreibung entwickeln. Es basiert konzeptionell auf dem in Abschnitt 2.6.3 vorgestellten Ansatz unterer Kollisionsschranken. Die Berechnung frühster Kollisionszeitpunkte erfordert Informationen über den zeitpunktbezogenen Abstand der Körper. Die Lösung des statischen Abstandsberechnungsproblems ist Gegenstand dieses Kapitels. Ein Überblick über die bisherigen Ansätze sowie die Vorstellung ihrer Stärken und Schwächen motiviert die Verwendung konservativer und hierarchisch verfeinerter Körperapproximationen, sogenannte Hüllkörperhierarchien. Diese erlauben eine drastische Reduktion der erforderlichen elementaren Abstandsberechnungen zwischen den Oberflächenelementen des Körpers. Neben der effizienten Ausgestaltung dieser Elementaroperationen müssen wir Verfahren zur Bestimmung von Hüllkörpern optimaler Approximationsgüte sowie zum Aufbau der Hüllkörperhierarchie vorstellen. Zwei weitere Schwerpunkte sind die Berechnung der Abstände zwischen Hüllkörperpaaren und die Entwicklung einer intelligenten Durchmusterungsstrategie für die Hüllkörperhierarchien zweier Körper. Durch die Verwendung der von uns entwickelten Beschleunigungsansätze kann das Abstandsberechnungsverfahren in der Praxis weiter beschleunigt werden.

3.1 Körperabstand und naheste Oberflächenelemente

Wir betrachten einen Körper als eine Punktmenge $\mathcal{K} \neq \emptyset$, die kompakt, zusammenhängend und starr ist. Aufgrund der Körperbewegung ist die Punktmenge einer zeitlichen Veränderung im Simulationsintervall $[0, T]$ unterworfen. Die Lage der Punktmenge $\mathcal{K}$ zum Zeitpunkt $t \in [0, T]$ wird daher folgendermaßen beschrieben:

$$\mathcal{K}(t) = \left\{ \mathbf{x}(t) \,\middle|\, x \in \mathcal{K} \right\}, \quad \text{mit } \mathcal{K} \equiv \mathcal{K}(0) \,.$$

Wir wollen die zeitliche Parametrisierung im folgenden nicht mehr explizit mitführen, da wir den Körper zu einem festen Zeitpunkt $t \in [0, T]$ betrachten. In Kapitel 4, in dem die Abstandsberechnung zu einer dynamischen Kollisionserkennung ausgebaut wird, werden wir jedoch daran erinnert, daß die Körperpunktmengen sowie deren Abstände Funktionen der Zeit darstellen.

Aus der Euklidischen Norm eines Vektors wird der Euklidische Abstand zweier

Punkte p und q abgeleitet:

$$\delta(p,q) := \|\mathbf{p}-\mathbf{q}\| = \sqrt{\sum_{i=1}^{3}(\mathbf{p}_i-\mathbf{q}_i)^2}\,.$$

Es erweist sich gelegentlich als günstig den quadratischen Abstand $\delta^2(p,q) := \delta(p,q)^2$ zugrunde zu legen, da durch die monotone Transformation Wurzelterme eliminiert werden können.

Die Abstandsdefinition zweier Punkte kann nun auf beliebige Punktmengen P und Q erweitert werden.

$$\delta(P,Q) := \inf\left\{\delta(p,q)\,\middle|\, p\in P, q\in Q\right\}.$$

Man beachte, daß die Distanz den Wert 0 annimmt, falls P und Q nicht disjunkt sind. Schließlich wollen wir noch endliche Mengen beliebiger Punktmengen als Argumente der Abstandsfunktion zulassen. Seien $\mathcal{P} = \{P_1,\ldots,P_n\}$ und $\mathcal{Q} = \{Q_1,\ldots,Q_m\}$ solche Mengen, dann definieren wir:

$$\delta(\mathcal{P},\mathcal{Q}) = \min\left\{\delta(P_i,P_j)\,\middle|\, 1\le i\le n, 1\le j\le m\right\}.$$

Um die Euklidische Distanzdefinition zwischen den verschiedenen Argumenttypen wohlzudefinieren, müssen wir die folgenden Vereinbarungen treffen. Sei p ein Punkt im $\mathbb{R}^3$, P eine Punktmenge und $\mathcal{P}$ eine Menge von Punktmengen, dann gelte:

$$\begin{aligned}
\delta(p,P) &= \min_{q\in P}\delta(p,q)\,,\\
\delta(p,\mathcal{P}) &= \min_{Q\in\mathcal{P}}\delta(p,Q)\,,\\
\delta(P,\mathcal{P}) &= \min_{Q\in\mathcal{P}}\delta(P,Q)\,.
\end{aligned}$$

Aufgrund der Eigenschaften, die wir für die zu simulierenden Körper abgeleitet haben, gilt für den *minimalen Euklidischen Abstand zweier Körper* $\mathcal{K}_1$, $\mathcal{K}_2$:

$$\delta(\mathcal{K}_1,\mathcal{K}_2) = \min\left\{\delta(x_1,x_2)\,\middle|\, \mathbf{x}_i\in\mathcal{K}_i,\ i=1,2\right\}.$$

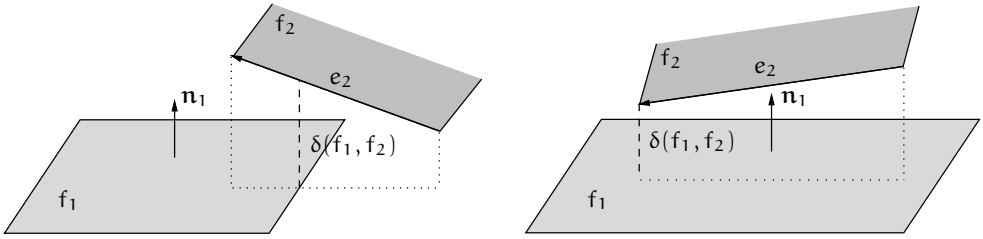
Der hier verwendete Abstandsbegriff umfaßt im Gegensatz zu den Definitionen in [Cam97] und [Mir97b] kein Maß für die Durchdringungstiefe zweier Körper. Dies liegt in der Tatsache begründet, daß sich die entsprechenden Abstandsgrößen lediglich im Fall konvexer Körper effizient berechnen lassen.

Zur Bestimmung von $\delta(\mathcal{K}_1,\mathcal{K}_2)$ erweist sich die folgende Beobachtung als hilfreich:

Beobachtung 1. Für zwei starre Körper $\mathcal{K}_1$ und $\mathcal{K}_2$ gilt:

$$\delta(\mathcal{K}_1,\mathcal{K}_2) = \delta(\partial\mathcal{K}_1,\partial\mathcal{K}_2)\,.$$

Abbildung 3.1 Der Euklidische Abstand zweier disjunkter Flächen f_1 und f_2 wird an einem Kante-Kante- bzw. einem Punkt-Fläche-Paar angenommen.



Kante-Kante-Fall Punkt-Fläche-Fall

Dies impliziert, daß wir uns zur Bestimmung der Distanz zweier Körper auf die Ermittlung des minimalen Abstands der Randdarstellungen zurückziehen dürfen. Falls $\mathcal{F}_1$ und $\mathcal{F}_2$ die Flächenmengen der Körper $\mathcal{K}_1$, $\mathcal{K}_2$ bezeichnen, folgt aus Beobachtung 1 für den minimalen Abstand des Körpers:

$$\delta(\mathcal{K}_1, \mathcal{K}_2) = \min\left\{\delta(f_1, f_2) \,\middle|\, f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2\right\}.$$

Bisher haben wir lediglich die Modellannahmen verwendet, die in die Definition des starren Körpers eingegangen sind und sich unmittelbar aus den Eigenschaften realer Körper ableiten ließen. Es ist offensichtlich, daß an dieser Stelle die geometrische Repräsentation des Körperrandes eine Rolle spielt. Wird die Oberfläche nämlich durch Polygone beschrieben so gilt, daß der minimale Abstand zweier disjunkter Flächen f_1, f_2 stets in einem *Punkt-Fläche-Paar* oder einem *Kante-Kante-Paar* der beiden Polygone angenommen wird (vgl. Abbildung 3.1).

$$\delta(f_1, f_2) = \begin{cases} \min\left\{\delta\big(f_1, \mathcal{V}(f_2)\big),\ \delta\big(\mathcal{V}(f_1), f_2\big),\ \delta\big(\mathcal{E}(f_1), \mathcal{E}(f_2)\big)\right\} & \text{falls } f_1 \cap f_2 = \emptyset\,; \\ 0 & \text{sonst}\,, \end{cases} \tag{3.1}$$

wobei $\mathcal{V}(f_i)$ die Menge der Eckpunkte und $\mathcal{E}(f_i)$ die Kantenmenge der Fläche f_i, $i = 1, 2$ bezeichnet.

Die Frage, ob die beiden Flächen f_1 und f_2 sich überlappen, läßt sich auf einen Test zwischen allen Kante-Fläche-Paaren der beiden Polygone zurückführen:

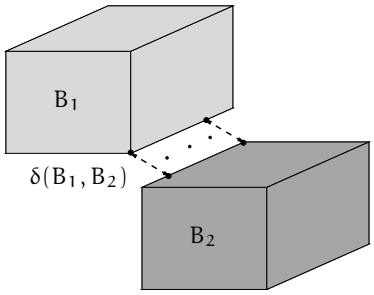
$$f_1 \cap f_2 \neq \emptyset \quad \Longleftrightarrow \quad (\exists e_1 \in f_1 :\ e_1 \cap f_2 \neq \emptyset) \ \vee\ (\exists e_2 \in f_2 :\ e_2 \cap f_1 \neq \emptyset)$$

Wir erhalten somit das folgende Lemma:

Lemma 3.1. *Für zwei starre Körper $\mathcal{K}_1$, $\mathcal{K}_2$, die als Polyeder mit Flächenmenge $\mathcal{F}_i$, Kantenmenge $\mathcal{E}_i$ und Knotenmenge $\mathcal{V}_i$, $i = 1, 2$ modelliert sind, gilt:*

$$\delta(\mathcal{K}_1, \mathcal{K}_2) = \begin{cases} \min\left\{\delta(\mathcal{F}_1, \mathcal{V}_2)\,, \delta(\mathcal{V}_1, \mathcal{F}_2),\ \delta(\mathcal{E}_1, \mathcal{E}_2)\right\} & \textit{falls } \mathcal{K}_1 \cap \mathcal{K}_2 = \emptyset\,; \\ 0 & \textit{sonst}\,. \end{cases}$$

Abbildung 3.2 Das naheste Punktepaar zwischen zwei Körpern ist im allgemeinen nicht eindeutig.



Die Aufgabe der elementaren Abstandsberechnung im Kontext einer Dynamiksimulation beschränkt sich jedoch nicht nur auf die Berechnung von *Abstandsinformationen* zwischen Flächenpaaren. Zwar genügen diese Daten, um die durch die Objektbewegungen vorgegebene Abstandsfunktion zwischen einem Körperpaar zu diskretisieren und eventuell zu approximieren, doch ist es oft hilfreich und gelegentlich notwendig, ein Paar von *nahesten Oberflächenelemeneten* (*closest features*) zu identifizieren, die als Zeuge (*witness*) des ermittelten Abstandsminimums eintreten können. Im Fall eines Kontakts werden diese Informationen benötigt, um mit Hilfe von in der Kontaktregion angreifenden Kräften oder Impulsen die Kollision physikalisch korrekt abwickeln zu können. Ist die Positionierung der Flächenmenge dagegen kollisionsfrei, so kann eine Speicherung der nahesten Oberflächenelemente im Rahmen einer *Caching*-Strategie zur Beschleunigung der Abstandsberechnung eingesetzt werden (*closest-features tracking*). Die Punkte minimalen Abstands auf beiden Körpern sind im allgemeinen nicht eindeutig, wie Abbildung 3.2 verdeutlicht. Es stellt sich daher die Frage, ob im Rahmen der Dynamiksimulation die Menge aller nahesten Punkte bestimmt werden müssen, oder ob es genügt, sich auf ein Punktepaar als *Repräsentant des minimalen Abstands* zu beschränken. Solange die Körper nicht miteinander in Kontakt treten, ist im allgemeinen die vollständige Menge der nahesten Punkte nicht von Interesse. Die Kollisionserkennungsfrage ist bereits durch die Identifikation eines einzigen nahesten Punktepaares beantwortet. Dieser Zeuge des Abstandsminimums wird sich auch im Rahmen unserer *Caching*-Strategie, die wir in Abschnitt 3.9 vorstellen werden, als hilfreich erweisen. Erst zum Zeitpunkt der Kollision ist es erforderlich, alle Kontaktregionen (Punkte, Kanten und Flächen) zu bestimmen, um Mehrfachkontakte korrekt abwickeln zu können. Die impulsbasierte Simulationsmethode, die unserer Arbeit zugrunde liegt, betrachtet jedoch ausschließlich Einpunktkontakte, so daß es auch im Rahmen der Kollisionsauflösung genügt, lediglich ein nahestes Punktepaar zu bestimmen. Wir wollen in diesem Kapitel eine Funktion χ definieren, die einem Paar (F_1, F_2) von Oberflächenelementen der beiden Körper einen Vertreter der Menge aller nahester Punktepaare zwischen F_1 und F_2 zuordnet. Dabei bezeichnet $\chi_1(F_1, F_2)$ und $\chi_2(F_1, F_2)$ die erste bzw. zweite Komponente des Punktepaares. Das naheste Punktepaar der beiden Körper, das wir

in Analogie mit $\chi(\mathcal{K}_1, \mathcal{K}_2)$ bezeichnen wollen, kann dabei beliebig aus der Menge der nahesten Punktepaare aller Oberflächenelemente der beiden Körper gewählt werden. Nach Lemma 3.1 gilt somit:

$$\chi(\mathcal{K}_1, \mathcal{K}_2) \in \begin{cases} \{\chi(F_1, F_2) \,|\, \delta(F_1, F_2) = \delta(\mathcal{K}_1, \mathcal{K}_2), (F_1, F_2) \in \\ \qquad \mathcal{F}_1 \times \mathcal{V}_2 \cup \mathcal{V}_1 \times \mathcal{F}_2 \cup \mathcal{E}_1 \times \mathcal{E}_1 \,\} & \text{falls } \mathcal{K}_1 \cap \mathcal{K}_2 = \emptyset\,; \\ \{\chi(F_1, F_2) \,|\, F_1 \cap F_2 \neq \emptyset,\ (F_1, F_2) \in \\ \qquad \mathcal{E}_1 \times \mathcal{F}_2 \cup \mathcal{F}_1 \times \mathcal{E}_2 \quad \} & \text{sonst}\,. \end{cases}$$

3.2 Verfahren für konvexe Polyeder

Der Spezialfall konvexer Polyeder ist in der Literatur intensiv diskutiert worden [DK83, GJK88, Bar92, LC91, Lin93, Cam97, Mir97b]. Neben den theoretischen Ergebnissen, die wir in 2.6.2 vorgestellt haben, erlaubt diese Einschränkung der Körpergeometrie den Einsatz äußerst effizienter Algorithmen von großer praktischer Bedeutung. Wie wir bereits in 2.6.4 erwähnt haben, lassen sich diese Algorithmen in zwei Kategorien einteilen, nämlich *featurebasierte* und *simplexbasierte* Verfahren. Allen gemeinsam ist die Fähigkeit, unter Ausnutzung *temporärer Kohärenz* sublineare Laufzeiten zu erzielen.

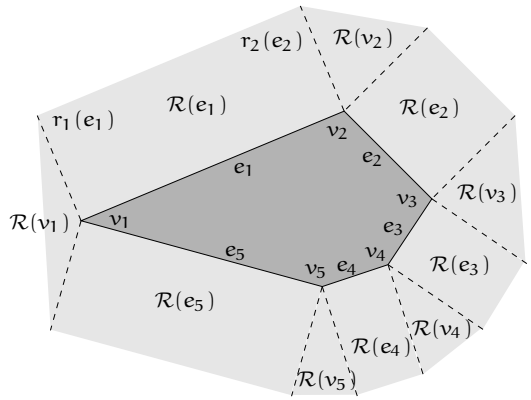
3.2.1 Featurebasierte Verfahren

Die *Oberflächenelemente* eines Polyeders sind die Eckpunkte, Kanten und Flächen, anhand derer die Oberfläche des Polyeders beschrieben wird. Featurebasierte Algorithmen führen geometrische Operationen auf den Oberflächenelementen durch, und zwar mit dem Ziel das Kollisionserkennungs- bzw. Abstandsberechnungsproblem zu lösen. Über lange Zeit galt der *closest-features tracking-Algorithmus* von LIN und CANNY [LC91, Lin93] als das effizienteste Verfahren zur Bestimmung des Abstands zweier konvexer Polyeder. Erst die Weiterentwicklung durch MIRTICH, die zu dem sogenannten V-Clip-Algorithmus [Mir97b] führte, konnte die empirische Laufzeit des Lin-Canny-Verfahrens verbessern. Der V-Clip-Algorithmus verhält sich zudem in degenerierten geometrischen Situationen unproblematischer und zeichnet sich neben einer höheren numerischen Stabilität durch einfachere Implementierbarkeit aus.

Der closest-features tracking-Algorithmus

Das von LIN und Canny [LC91, Lin93] vorgestellte Verfahren läuft ausgehend von einem Paar von Oberflächenelementen der beiden Körper über deren Randdarstellungen und tauscht anhand lokaler Tests die Oberflächenelemente durch benachbarte Elemente aus, bis keine Verbesserung des Distanzwertes mehr erzielt werden kann. Wir wollen den Algorithmus für den zweidimensionalen Fall skizzieren. Die Übertragung in den $\mathbb{R}^3$ ist offensichtlich und kann in [Lin93] im Detail nachgelesen werden. Im Mittelpunkt des Lin-Canny-Algorithmus' steht der Begriff der *externen Voronoiregion* eines Oberflächenelementes.

Abbildung 3.3 Die externen Voronoiregionen eines Polygons.



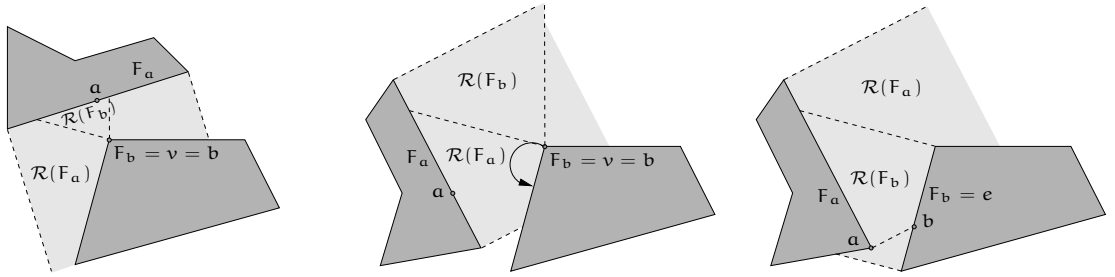
Definition 3.1 (Externe Voronoiregion eines Oberflächenelementes). Sei F ein Oberflächenelement eines d-dimensionalen Körpers im Sinne von Definition 2.1. Die Voronoiregion von F, die wir mit $\mathcal{R}_V(F)$ bezeichnen wollen, ist die Menge aller Punkte außerhalb des Körpers, die zu F näher liegen, als zu allen anderen Oberflächenelementen des Körpers. □

Im $\mathbb{R}^2$ betrachten wir als Körper ein Polygon, dessen Oberfläche durch Eckpunkte und Kanten beschrieben wird. Abbildung 3.3 zeigt einen solchen zweidimensionalen Körper sowie die Voronoiregionen seiner Oberflächenelemente, die den Raum außerhalb des Polygons partitionieren. Die Voronoiregionen können auf einfache Weise bestimmt werden. Dazu betrachtet man für jeden Eckpunkt v des Polygons die beiden von ihm ausgehenden Strahlen $r_1(v)$ und $r_2(v)$, deren Richtungsvektoren senkrecht auf den zu v inzidenten Kanten stehen. Diese Strahlen grenzen die Voronoiregionen der Oberflächenelemente voneinander ab. Die Region eines Eckpunktes v ist der unendliche Kegel, der von den beiden Strahlen $r_1(v)$ und $r_2(v)$ eingeschlossen wird. Als Voronoiregion einer Kante erhält man das in einer Richtung offene Rechteck, das durch die Kante und die beiden auf ihr senkrecht stehenden Strahlen gebildet wird. Der Lin-Canny Algorithmus basiert auf dem folgenden Satz, der in [Lin93] für den dreidimensionalen Fall bewiesen wird.

Satz 3.2. *Seien* A *und* B *zwei disjunkte, konvexe Polygone und* a *und* b *die nahesten Punkte zwischen den Oberflächenelementen* F_a *von* A *bzw.* F_b *von* B. *Dann gilt, daß* a *und* b *naheste Punkte zwischen* A *und* B *darstellen, wenn* $a \in \mathcal{R}_V(F_b)$ *und* $b \in \mathcal{R}_V(F_a)$ *ist.*

Um zu verstehen, wie dieses hinreichende Kriterium zur Bestimmung des nahesten Paares von Oberflächenelementen benutzt werden kann, betrachten wir die Situation im linken Teil von Abbildung 3.4. Die nahesten Punkte a und b der Oberflächenelemente F_a und F_b liegen jeweils in der Voronoiregion des anderen Elementes. Nach Satz 3.2 handelt es sich also um ein nahestes Punktepaar der beiden Polygone A und B. Der

Abbildung 3.4 Die Arbeitsweise des Lin-Canny-Algorithmus: Nach der Objektbewegung (Bild 2) wird das naheste Oberflächenelement F_b aktualisiert (Bild 3).



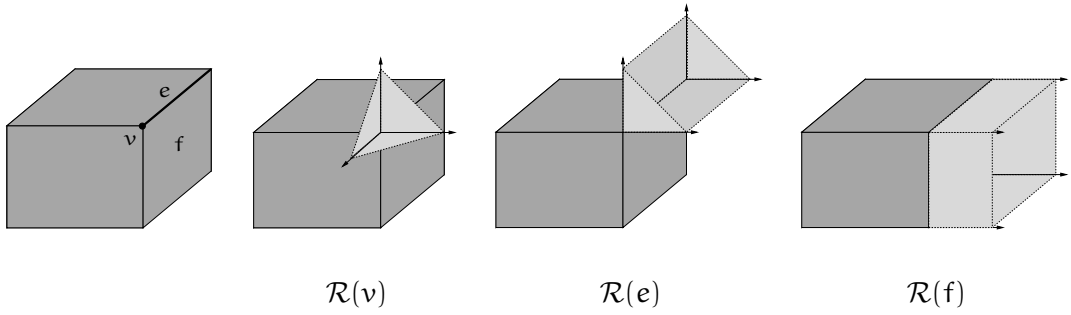
$b \in \mathcal{R}(F_a)$ und $a \in \mathcal{R}(F_b)$ $b \in \mathcal{R}(F_a)$ aber $a \notin \mathcal{R}(F_b)$ $b \in \mathcal{R}(F_a)$ und $a \in \mathcal{R}(F_b)$

so gefundene Polygonabstand kann nicht mehr verbessert werden, das Verfahren terminiert. Das in dem rechten Teil von Abbildung 3.4 dargestellten Szenario bezeichnen a und b ebenfalls die Punkte minimalen Abstands zwischen F_a und F_b. Zwar liegt b immer noch in der Voronoiregion von F_a, doch a hat die Region von F_b verlassen. Dies erkennt man daran, daß a auf der „falschen" Seite des Voronoistrahls $r_1(v)$ liegt. Der Lin-Canny-Algorithmus aktualisiert nun das Oberflächenelement F_b. Dieses wechselt von dem Knoten v zu der zu ihm inzidenten Kante e. Anschließend werden die nahesten a, b zwischen F_a und F_b neu berechnet und der *Voronoi-Test* wiederholt. Das Paar von Oberflächenelementen wird nun solange aktualisiert, bis ein Paar nahester Punkte zwischen den beiden Körpern gefunden ist.

Die Korrektheit sowie die Terminierung des Verfahrens, wobei letztere nur unter bestimmten Voraussetzungen garantiert ist, werden in [Lin93] bewiesen. Dort findet man auch eine detaillierte Beschreibung der Vorgehensweise im dreidimensionalen Fall. Der wesentliche Unterschied beschränkt sich dabei jedoch auf die Gestalt der Voronoiregionen, die durch Ebenen statt Strahlen begrenzt sind (vgl. Abbildung 3.5).

Für die Laufzeit des Verfahrens ist *temporäre Kohärenz* von großer Bedeutung. Das naheste Paar von Oberflächenelementen ändert sich nur selten, so daß es Sinn macht das Paar des letzten Zeitschritts zwischenzuspeichern und als Ausgangspunkt im nächsten Aufruf der Abstandsberechnung zu verwenden. Selbst wenn die Körper sich sehr schnell bewegen oder ihr Polyedermodell sehr fein aufgelöst ist, stellt das Paar von Oberflächenelementen des letzten Zeitschritts eine günstige Initialisierung des Lin-Canny-Verfahrens dar. Diese Ausnutzung temporärer Kohärenz macht die Laufzeit des Algorithmus' nahezu unabhängig von der Polyederkomplexität, so daß man dem Verfahren in der Praxis eine konstante Laufzeit bescheinigt [Mir97a]. In [CLMP95] wurde jedoch gezeigt, daß die erforderliche Rechenzeit bei nicht vorhandener Kohärenz etwa linear mit der Beschreibungskomplexität der Polyeder wächst.

Abbildung 3.5 Die externen Voronoiregionen des Eckpunktes v, der Kante e und der Fläche f eines achsenorientierten Quaders.



Der V-Clip-Algorithmus

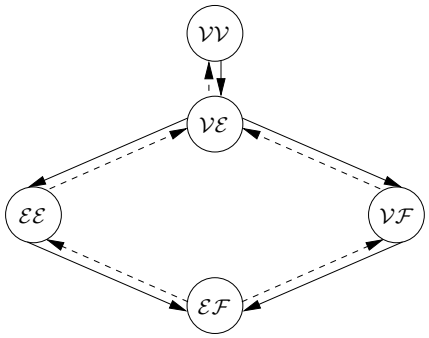
Der Lin-Canny-Algorithmus weist trotz hoher Effizienz zwei wesentliche Schwachpunkte auf:

1. Im Fall von sich durchdringenden Polyedern gerät der Algorithmus in eine Endlosschleife. Um die Terminierung garantieren zu können, ist die Einführung eines künstlichen Abbruchkriteriums, basierend auf der Zahl der durchgeführten Iterationen, erforderlich. Eine Wahl für die Iterationenbegrenzung, die sich in allen Situationen als ausreichend erweist und die Laufzeit des Algorithmus' nicht allzu sehr beeinträchtigt, ist schwierig. Daneben macht das Verhalten des Algorithmus' im Kollisionsfall die Bestimmung eines Maßes für die Durchdringungstiefe der Polyeder unmöglich. Eine solche Information ist jedoch zur Ermittlung des Kollisionszeitpunktes mittels Nullstellensuche auf der Distanzfunktion hilfreich, wie wir in Abschnitt 2.6.1 erläutert haben.

2. Auch degenerierte geometrische Situationen können den Algorithmus in einen Zyklus führen. Ein großer Teil des Programmcodes setzt sich mit der korrekten Behandlung dieser degenerierten Fälle auseinander. Hinzu kommt, daß die numerische Stabilität durch die Einführung von Toleranzen beeinträchtigt wird.

An diesen Schwachpunkten setzen die Modifikationen von MIRTICH an, die zu dem sogenannten V-Clip-Algorithmus [Mir97b] geführt haben. Dieser basiert weiterhin auf der Grundidee des bereits vorgestellten Lin-Canny-Algorithmus'. Auch der V-Clip-Algorithmus sucht nach Oberflächenelementen F_a, F_b die das hinreichende Kriterium aus Satz 3.2 erfüllen. Ist die Bedingung für ein Oberflächenelement verletzt, so wird das Element solange durch Nachbarelemente ersetzt, bis die Distanz des aktuellen Paares von Oberflächenelementen (F_a, F_b) nicht mehr verringert werden kann. Ein entscheidender Unterschied zum Lin-Canny-Algorithmus besteht nun darin, daß die nahesten Punkte zwischen F_a und F_b bis zum Zeitpunkt der Terminierung niemals explizit berechnet werden. Die degenerierten geometrischen Situationen, die dem Lin-

Abbildung 3.6 Zustände und Übergänge des V-Clip-Algorithmus'.



Canny-Algorithmus Schwierigkeiten bereitet haben, erweisen sich aus diesem Grund als unproblematisch. Des weiteren ermöglicht die Vorgehensweise die Beschränkung des Verfahrens auf disjunkte Körper zu beseitigen. Der V-Clip-Algorithmus ist selbst im Kollisionsfall in der Lage, Abstandsinformationen in Form der Durchdringungstiefe zurückzuliefern.

Im Rahmen der Aktualisierung eines Oberflächenelementes werden zwei Fälle unterschieden. Beim Übergang auf ein *höherdimensionales* Element stellt das Verfahren sicher, daß die Distanz der Oberflächenelemente sinkt. Wird dagegen ein Oberflächenelement durch ein Element *niedrigerer Dimension* ersetzt, so garantiert die Vorgehensweise, daß der Abstand von F_a und F_b unverändert bleibt. Aus dem Zustandsdiagramm in Abbildung 3.6 können alle möglichen Übergänge zwischen Paaren von Oberflächenelementtypen abgelesen werden. $\mathcal{V}$ steht dabei für einen Eckpunkt, $\mathcal{E}$ für eine Kante und $\mathcal{F}$ für eine Fläche. Ein durchgezogener Pfeil markiert einen Übergang, der zu einer Verringerung des Abstands von F_a und F_b führt, während ein gestrichelter Pfeil eine Ersetzung symbolisiert, die die Distanz der Oberflächenelemente unverändert läßt. Jeder unendlich lange Weg in dem Graphen muß eine unendliche Zahl durchgezogener Pfeile benutzen, die alle zu einer Verringerung des Abstands zwischen F_a und F_b führen. Da es jedoch nur endlich viele Paare von Oberflächenelementen gibt, ist dies nicht möglich und eine Terminierung des Verfahrens garantiert.

Die Modifikationen von MIRTICH verhindern also das zyklische Verhalten im Kollisionsfall, so daß ein Maß für die Durchdringungstiefe ähnlich effizient wie im kollisionsfreien Fall berechnet werden kann. Degenerierte geometrische Situationen stellen für den Algorithmus kein Problem dar, was den Programmcode einfacher und den Verzicht auf numerische Toleranzen erst möglich macht.

3.2.2 Simplexbasierte Verfahren

Im Gegensatz zu den bisher vorgestellten Verfahren, in denen die Oberflächenelemente im Mittelpunkt stehen, operieren simplexbasierte Verfahren auf den *Simplices* der Polyeder. Diese sind durch Teilmengen der Eckpunktmenge bestimmt. Ein d-Simplex

des $\mathbb{R}^3$ ist die Konvexkombination von $d+1$ linear unabhängigen Punkten des $\mathbb{R}^3$. Die simplexbasierten Verfahren arbeiten somit auf Punkten, Geradensegmenten, Dreiecken und Tetraedern. Der *GJK-Algorithmus* [GJK88] ist der wichtigste Vertreter dieser Klasse von Abstandsberechnungsverfahren. Die Algorithmen von RABBITZ [Rab94] und CAMERON [Cam97] führen Modifikationen zur Verbesserung der praktischen Laufzeit durch.

Der GJK-Algorithmus

In [GJK88] haben GILBERT, JOHNSON und KEERTHI einen Algorithmus vorgestellt, der in „erwarteter" Laufzeit $O(|\mathcal{V}(P)| + |\mathcal{V}(Q)|)$ den Abstand zweier konvexer Polyeder P und Q bestimmt. Im Überlappungsfall liefert das Verfahren als Maß für die Durchdringungstiefe die minimale Länge aller Verschiebungsvektoren, die die beiden Polyeder voneinander trennen (*minimale translatorische Distanz*). Der Algorithmus betrachtet dazu die *sogenannte Minkowski-Differenz* $Q - P$ der beiden konvexen Polyeder.

$$Q - P := \left\{ \mathbf{q} - \mathbf{p} \,\middle|\, q \in Q,\, p \in P \right\}.$$

Das Resultat dieser Operation ist selbst wieder ein konvexer Polyeder, das sogenannte *Minkowski-Differenz-Polyeder* Z. Die folgende Beobachtung zeigt die Bedeutung von Z im Rahmen der Abstandsberechnung:

$$\min_{z \in Z} \|z\| = \begin{cases} \delta(P, Q) & \text{falls } P \cap Q = \emptyset\,; \\ \min\left\{ \|\mathbf{q} - \mathbf{p}\| \,\middle|\, q \in Q,\, p \in P,\, P \cap \big(Q + (\mathbf{q} - \mathbf{p})\big) = \emptyset \right\} & \text{falls } P \cap Q \neq \emptyset\,. \end{cases}$$

Dabei bezeichnet $Q + \mathbf{t}$ das Ergebnis der Anwendung einer Translation $\mathbf{t}$ auf alle Punkte der Punktemenge Q.
Durch die Betrachtung der Minkowski-Differenz wird das Kollisionserkennungsproblem auf die Beantwortung der Frage reduziert: Ist der Ursprung o in Z enthalten? Es gilt nämlich:

$$\begin{aligned} P \cap Q = \emptyset \quad &\Longleftrightarrow \quad \exists\, p \in P, q \in Q : \ \mathbf{p} = \mathbf{q} \\ &\Longleftrightarrow \quad \exists\, p \in P, q \in Q : \ \mathbf{p} - \mathbf{q} = \mathbf{0} \\ &\Longleftrightarrow \quad o \in Q - P = Z\,. \end{aligned}$$

Der GJK-Algorithmus sucht nun einen Simplex von Z, der den geringsten Abstand zum Ursprung besitzt (*Abstandsberechnung*) bzw. der den Ursprung enthält (*Kollisionserkennung*). Dabei wird das Minkowski-Differenz-Polyeder jedoch nicht explizit berechnet. Aufgrund der Konvexität von Z genügt es die benötigten Eckpunkte bei Bedarf zu bestimmen. Durch die nahesten Punkte von P und Q verläuft jeweils eine unterstützende Ebene, die senkrecht auf dem Verbindungsvektor dieser Punkte steht. Für diese unterstützenden Ebenen gilt, daß das entsprechende konvexe Polyeder vollständig auf einer Seite der zu ihm gehörenden Ebene liegt. Die Existenz eines solchen Paares unterstützender Ebenen durch Punkte p und q ist ein *notwendiges und hinreichendes Kriterium* dafür, daß man anhand von p und q (und deren lokalen Umgebungen) den Abstand

von P und Q bestimmen kann. Überträgt man das Kriterium auf das Polyeder Z, so wird das Abstandsminimum durch einen Punkt $w \in Z$ definiert, dessen unterstützende Ebene mit Normalenvektor $\boldsymbol{w}$ alle Punkte aus Z vom Ursprung trennt.

Der Algorithmus startet nun mit einem beliebigen Simplex von Z. Im einfachsten Fall handelt es sich um einen 0-Simplex, der durch ein beliebiges Eckpunktpaar (q, p) der beiden Polyeder definiert wird. In einem ersten Schritt ist der naheste Punkt w des Simplex vom Ursprung zu bestimmen (im Fall des 0-Simplex: $\boldsymbol{w} = \mathbf{q} - \mathbf{p}$). Die unterstützende Ebene Σ_w, die auf dem Ortsvektor $\boldsymbol{w}$ von w senkrecht steht, ist durch die Nullstellen der folgenden Funktion gegeben:

$$\begin{aligned} g_w : \mathbb{R}^3 &\to \mathbb{R} \\ \boldsymbol{z} &\mapsto \boldsymbol{w}^T\boldsymbol{w} - \boldsymbol{w}^T\boldsymbol{z} \,. \end{aligned}$$

1. Fall: $\forall v \in \mathcal{V}(Z) : \; g_w(\boldsymbol{v}) \leq 0$
   Das konvexe Polyeder Z liegt bezüglich Σ_w vollständig auf der dem Ursprung gegenüberliegenden Seite, da diese Bedingung für alle Eckpunkte erfüllt ist. Wir haben daher das gesuchte Punktepaar (q, p) gefunden, denn $\delta(q, p) = \|\boldsymbol{w}\|$ ist der minimale translatorische Abstand der beiden Polyeder.

2. Fall: $\exists v \in \mathcal{V}(Z) : \; g_w(\boldsymbol{v}) > 0$
   In diesem Fall stellen die w charakterisierenden Punkte nicht das gesuchte Punktepaar dar. Aufgrund der Konvexität gilt, daß das Geradensegment zwischen w und v vollständig innerhalb von Z liegt, d.h.

   $$\boldsymbol{x}(\lambda) = \boldsymbol{w} + \lambda(\boldsymbol{v} - \boldsymbol{w}) \in Z \,, \quad 0 \leq \lambda \leq 1 \,.$$

   Für den quadratischen Abstand des Punktes $x(\lambda)$ zum Ursprung gilt:

   $$\delta^2\big(x(\lambda), o\big) = \boldsymbol{x}(\lambda)^2 = \big[\boldsymbol{w} + \lambda(\boldsymbol{v} - \boldsymbol{w})\big]^2 \,.$$

   Die Ableitung der Abstandsfunktion in dem Parameterwert $\lambda = 0$ liefert:

   $$\begin{aligned} \frac{d\delta^2(x(\lambda), o)}{d\lambda}(0) &= -2\boldsymbol{w}^T\boldsymbol{w} + 2\boldsymbol{w}^T\boldsymbol{v} \\ &= -2g_w(\boldsymbol{v}) < 0 \,. \end{aligned}$$

   Es existieren also Punkte auf dem Geradensegment durch w und v, die näher zum Ursprung liegen als w.

Die Funktion g_w liefert einen einfachen Optimalitätstest für den Punkt $w \in Z$. Man sucht einen Eckpunkt (Konvexität!) $v_w \in Z$ mit

$$g_w(\boldsymbol{v}_w) = \max_{v \in \mathcal{V}(Z)} g_w(\boldsymbol{v}) = \max_{v \in \mathcal{V}(Z)} -\boldsymbol{w}^T\boldsymbol{v} \,.$$

Der Punkt v_w wird als *unterstützender Eckpunkt von Z in Richtung von* $\boldsymbol{w}$ bezeichnet. Die Bestimmung von v_w kann auf die Suche nach unterstützenden Eckpunkten q_w, p_w von Q und P zurückgeführt werden.

$$\max_{v \in \mathcal{V}(Z)} -\boldsymbol{w}^T\boldsymbol{v} = \max_{q \in \mathcal{V}(Q), p \in \mathcal{V}(P)} -\boldsymbol{w}^T(\mathbf{q} - \mathbf{p}) = \max_{p \in \mathcal{V}(P)} \boldsymbol{w}^T\mathbf{p} + \max_{q \in \mathcal{V}(Q)} -\boldsymbol{w}^T\mathbf{q} \,.$$

Wir müssen anschließend die folgende Fallunterscheidung durchführen:

1. Fall: $g_w(\boldsymbol{v}_w) \leq 0$
   Der Punkt w mit Ortsvektor $\boldsymbol{w} = \mathbf{q} - \mathbf{p}$ bestimmt das gesuchte Punktepaar sowie die minimale translatorische Distanz $\delta(q, p) = \|\boldsymbol{w}\|$ der beiden konvexen Polyeder Q und P.

2. Fall: $g_w(\boldsymbol{v}_w) > 0$
   Der Punkt w verletzt das Optimalitätskriterium der trennenden Ebene, so daß der zugrundeliegende Simplex S_w verbessert werden muß. Man berechnet dazu die kleinste Teilmenge von Knoten aus S_w, die einen Simplex definieren, in dem w enthalten ist. Diese Teilmenge besteht aus mindestens einem und höchstens drei Eckpunkten von Z. Anschließend wird der Punkt v_w zu dieser Menge hinzugenommen. Das Resultat ist eine zwei- bis vierelementige Eckpunktmenge, die einen Simplex $S_{w'}$ definiert und alle Punkte auf dem Geradensegment zwischen w und v enthält.

Bei den Simplices handelt es sich um einfache geometrische Strukturen, so daß der naheste Punkt zum Ursprung, w', ohne großen Aufwand für die nächste Iteration berechnet werden. Da die Zahl der Simplices endlich ist, terminiert das Verfahren. In der Praxis erweist sich die Zahl der Iterationen als nahezu unabhängig von der Komplexität der zugrundeliegenden Polyeder. Die Bestimmung von v_w verursacht Kosten in Höhe von $O(|P| + |Q|)$. In der Praxis lassen sich daher lineare Laufzeiten erwarten.

Die modifizierten GJK-Algorithmen

Da die Zahl der Simplexverbesserungen als konstant erachtet werden kann, sind lediglich Modifikationen bezüglich der Vorgehensweise zur Bestimmung unterstützender Eckpunkte sinnvoll. Im Algorithmus von CHUNG TAT LEUNG [Leu96] wird eine *hierarchische Repräsentation* des konvexen Polyeders durch Simplices verwendet, die in der Arbeit von DOBKIN und KIRKPATRICK [DK83] vorgestellt wurde. Diese Datenstruktur erlaubt den unterstützenden Eckpunkt des konvexen Polyeders P in einer vorgegebenen Richtung in Zeit $O(\log|\mathcal{V}(P)|)$ zu bestimmen. Die Konstruktion verbessert die bei Kohärenz zu beobachtende „empirische " Laufzeit auf $O(\log|\mathcal{V}(P)| + \log|\mathcal{V}(Q)|)$, wobei P und Q die betrachteten konvexen Polyeder bezeichnen.

Von größerer praktischerer Bedeutung ist der Einsatz des sogenannten *Hill-Climbing-Verfahrens*. Zur Ermittlung des unterstützenden Knotens in Richtung $\boldsymbol{w}$ startet es mit einem beliebigen Knoten $v \in \mathcal{V}(P)$. Der Eckpunkt v wird nun mit allen Nachbarknoten v', d.h. mit allen durch eine Kante mit v verbundenden Eckpunkten hinsichtlich $\boldsymbol{w}^T\boldsymbol{v} < \boldsymbol{w}^T\boldsymbol{v}'$ verglichen. Ist diese Bedingung erfüllt, so setzt man v auf v' und führt wieder lokale Vergleichsoperationen mit den Nachbarknoten des aktualisierten Eckpunktes v durch. Auf diese Weise wandert das Verfahren über den Rand des konvexen Polyeders, bis es schließlich keinen adjazenten Knoten mehr findet, der den aktuellen Wert $\boldsymbol{w}^T\boldsymbol{v}$ verbessert. Die Konvexität des Polyeders garantiert dabei die Terminierung

des Verfahrens. Die Wahl des Startknotens beeinflußt nicht die Korrektheit des Algorithmus', wohl aber dessen Laufzeit. Hier haben sich, wie bereits beim Lin-Canny-Algorithmus 3.2.1 beschrieben, Caching-Strategien zur Ausnutzung temporärer Kohärenz bewährt. Zwischen zwei Aufrufen des Verfahrens bietet sich das Caching des optimalen Simplex aus dem letzten Test der beiden Polyeder an, während innerhalb eines einzelnen Aufrufs, d.h. zwischen zwei Simplexverbesserungen, das Zwischenspeichern des letzten unterstützenden Eckpunktes sinnvoll erscheint. Ein Kohärenzargument rechtfertigt diese Vorgehensweise, da sowohl die nahesten Simplices zwischen zwei Zeitschritten als auch die Richtungen $\boldsymbol{w}$ zwischen zwei Iterationen eines Aufrufs nur geringfügig abweichen. Falls die Kohärenzannahme zutrifft, läßt sich für den modifizierten GJK-Algorithmus eine empirisch konstante Laufzeit [Mir97b] erwarten.

3.3 Allgemeine Verfahren

Selbstverständlich dürfen wir zur Dynamiksimulation realer Körper nicht auf solch drastische Einschränkungen der Körpergeometrie zurückgreifen, wie sie von den Verfahren des vorangegangenen Abschnitts vorausgesetzt werden. Es ist daher unumgänglich, die vorgestellten Algorithmen für konvexe Polyeder auf die Abstandsberechnung allgemeiner Polyeder zu erweitern bzw. generelle Verfahren zu entwickeln, die keine einschränkenden Annahmen über die Geometrie der Körper machen.

3.3.1 Erweiterung der Verfahren für konvexe Polyeder

Unsere Überlegungen aus Abschnitt 3.1 haben gezeigt, daß zur Abstandsberechnung zweier Körper $\mathcal{K}_1, \mathcal{K}_2$ das folgende Optimierungsproblem gelöst werden muß:

$$\begin{aligned} \min\ & \delta(\mathcal{K}_1, \mathcal{K}_2) = \sum_{i=1}^{3} (x_{1i} - x_{2i})^2 \\ \text{s.d.}\quad & \mathbf{x}_1 \in \mathcal{K}_1,\, \mathbf{x}_2 \in \mathcal{K}_2 \end{aligned}$$

Falls der Körper $\mathcal{K}_1$ in r konvexe Teile $\mathcal{K}_{11}, \ldots, \mathcal{K}_{1r}$ zerlegt ist (vgl. Abschnitt 2.6.4) und Körper $\mathcal{K}_2$ als Vereinigung von s konvexen Polyedern $\mathcal{K}_{21}, \ldots, \mathcal{K}_{2s}$ dargestellt wird, läßt sich das allgemeine Abstandsberechnungsproblem auf den Fall für konvexe Polyedermodelle reduzieren. Dazu sind rs Probleme der quadratischen Programmierung mit linearen Nebenbedingungen zu lösen:

$$\begin{aligned} \min\quad & \delta(\mathcal{K}_{1i}, \mathcal{K}_{2j}) = \textstyle\sum_{k=1}^{3} (x_{1k} - x_{2k})^2 \\ \text{s.d.}\quad & \mathbf{x}_1 \in \mathcal{K}_{1i},\, \mathbf{x}_2 \in \mathcal{K}_{2j}\,, \quad 1 \le i \le r,\, 1 \le j \le s \end{aligned}$$

Neben den bekannten Algorithmen zur Lösung solcher Optimierungsaufgaben bieten sich insbesondere die effizienten Abstandsberechnungsverfahren aus Abschnitt 3.2 zur Lösung der Teilprobleme an.

In [PML95] wird ein Beschleunigungsansatz für diese Vorgehensweise vorgestellt. Dabei werden zunächst die *konvexen Hüllen* der beiden nichtkonvexen Polyeder und

statt der Distanz aller konvexen Teilkörperpaare der Abstand der konvexen Hüllen bestimmt. Der Abstand stellt eine untere Schranke für die tatsächliche Distanz der beiden Polyeder dar. Zur Beantwortung der Kollisionserkennungsfrage genügt diese Information, sofern der ermittelte Abstandswert größer Null ist. Im Fall einer Kollision der Hüllkörper ist man dagegen gezwungen, die Verpackung der beiden Körper aufzulösen und die rs Abstandsberechnungen durchzuführen. Das eigentliche Problem ist und bleibt jedoch die Bestimmung einer günstigen konvexen Zerlegung der Körper, was wir bereits in 2.6.4 diskutiert haben. Unter der Annahme, daß die Abstandsberechnungsverfahren aus Abschnitt 3.2 konstante Laufzeiten erzielen, wird die algorithmische Komplexität des Verfahrens für allgemeine Polyeder von der Zahl der konvexen Teilkörperpaare dominiert. Betrachtet man beispielsweise die Polyedermodelle zweier Tori, so beträgt die Laufzeit des Verfahrens $O(n^2)$, wobei n die Anzahl der Torusflächen beschreibt.

LIN, MANOCHA und PONAMGI haben in [PML95] versucht, das closest-features tracking-Verfahren für konvexe Polyeder auf die allgemeine Problemstellung zu übertragen, ohne auf eine konvexe Zerlegung der beteiligten Körper zurückgreifen zu müssen. In diesen Zusammenhang wurde der Lin-Canny Algorithmus derart erweitert, daß auch Überlappungen konvexer Polyeder erkannt werden. Das modifizierte closest-features tracking-Verfahren wird zunächst auf die konvexen Hüllen der beiden Polyeder angewendet. Bei dieser Vorgehensweise geht jedoch die exakte Abstandsinformation verloren, da im kollisionsfreien Fall nur die Distanz der konvexen Hüllen zurückgeliefert werden kann. Falls diese überlappen, unterscheidet man zwischen den kollidierenden Flächen der Originalpolyeder und denen, die erst durch die Konstruktion der konvexen Hülle entstanden sind. Letztere müssen gesondert behandelt werden, um die von ihnen überdeckten Kollisionsflächen der Originalobjekte finden zu können. Auf diese Weise wird die vollständige Kontaktregion bestimmt, was für die analytischen Verfahren der zwangsbasierten Kollisionsauflösung von Bedeutung ist. Allerdings muß die numerische Stabilität und somit auch die Praktikabilität des Verfahrens in Frage gestellt werden.

Aufgrund der Problematik im Zusammenhang mit der Bestimmung konvexer Zerlegungen wollen wir uns im folgenden mit Verfahren beschäftigen, die eine solche Vorverarbeitung nicht voraussetzen.

3.3.2 Hüllkörperbasierte Verfahren

Ähnlich wie im Fall der statischen Kollisionserkennung arbeiten auch alle hüllkörperbasierten Abstandsberechnungsverfahren mit einem einfachen Ausschlußtest für die im Rahmen der *Hüllkörperhierarchie* definierten Teile der beiden Objekte. Dabei nutzen sie die Tatsache aus, daß der Hüllkörperabstand eine untere Schranke für die Distanz der eingeschlossenen Teilkörper darstellt. Ist dieser Abstandswert größer oder gleich dem exakten Abstand, der zwischen zwei beliebigen Oberflächenelementen der Körper festgestellt wurde, so kann auf eine weitere Betrachtung des Teilkörpers verzichtet werden.

Die in Abschnitt 2.6.4 beschriebenen Vorteile der Verwendung von Hüllkörperhier-

archien, insbesondere die Fähigkeit mit unstrukturierten Flächenmengen sinnvoll umgehen zu können, prädestiniert die hüllkörperbasierten Abstandsberechnungsverfahren für den Einsatz in Dynamiksimulationen mit realem Anwendungshintergrund.

Im Gegensatz zu der Vielfalt der Hüllkörpertypen, die im Rahmen effizienter Ray-Tracing- und statischer Kollisionserkennungsverfahren studiert wurden, ist die Zahl der untersuchten Hüllkörpertypen im Rahmen des Abstandsberechnungsproblems vergleichsweise gering. Dies mag daran liegen, daß die Bestimmung der Distanz zwischen den in der Kollisionserkennung gebräuchlichen Hüllkörpertypen, wie beispielsweise beliebig orientierten Boxen, ungleich schwieriger ist, als die Beantwortung der Überlappungsfrage. Ein anderer Grund ist, daß der Abstandsberechnung generell nicht die Bedeutung zugemessen wird, die die Kollisionserkennung bereits erlangt hat. In vielen Anwendungen ist der inhärente Mehraufwand der Berechnung von Abstandsinformationen nicht erwünscht. Dennoch gibt es einige wenige Verfahren, die sich in dem von ihnen verwendeten Hüllkörpertyp unterscheiden. QUINLAN [Qui94] verwendet Kugeln als Hüllkörper, die LARSEN ET AL. [ESCD99] als sogenannte Swept-Sphere-Volumes verallgemeinern. Swept-Sphere-Volumes sind einfache geometrische Formen wie Punkte, Linien oder Rechtecke, die um einen konstanten „Radius“ ausgedehnt werden. So ist beispielsweise das Swept-Sphere-Volume eines Liniensegmentes ein Zylinder mit Kugelkappen. KONEČNÝ legt dem in [Kon98] vorgestellten Abstandsberechnungsverfahren Hüllkörper zugrunde, sie sich als Schnitt geeignet gewählter Halbebenen ergeben. JOHNSON und COHEN [DE98] verwenden beliebig orientierte Boxen, wobei sie zur Hüllkörperabstandsberechnung den simplexbasierten Algorithmus von CAMERON [Cam97] einsetzen. Dieser arbeitet jedoch lediglich auf relativ komplexen, konvexen Polyedern effizient. Achsenorientierte Boxen sind unverständlicherweise im Rahmen der hüllkörperbasierten Abstandsberechnung nicht betrachtet worden.

Wir wollen uns im Gegensatz zu obigen Verfahren nicht auf einen bestimmten Hüllkörpertyp festlegen. Statt dessen werden wir alle Hüllkörpertypen betrachten, die sich im Rahmen der statischen Kollisionserkennungsverfahren als effektiv erwiesen haben. Die zusätzlichen Aufgaben, die sich aus der Verwendung unterschiedlicher Hüllkörpertypen ergeben, betreffen nur am Rande den Aufbau der Hüllkörperhierarchie. Von größerer Bedeutung ist es, das Problem der approximationsgüteoptimalen Berechnung der verschiedenen Hüllkörper sowie das Hüllkörperabstandsberechnungsproblem effizient zu lösen.

Nach diesem Überblick über die bisherigen Lösungsansätze, werden wir unseren Ansatz zur Lösung des Abstandsberechnungsproblem erarbeiten. Dabei werden wir zunächst einen naiven, kombinatorischen Algorithmus diskutieren, der den Abstand zweier Körper auf der Basis von Lemma 3.1 durch Vollenumeration der Alternativenmenge löst. Anschließend werden wir zeigen, wie das naive Verfahren mit Hilfe der *Branch-and-Bound-Technik* beschleunigt werden kann.

3.3.3 Das naive Verfahren

Unsere Überlegungen in Abschnitt 3.1 legen einen einfachen Algorithmus zur Bestimmung des Euklidischen Abstands zweier Körper nahe. Gleichung 3.1 reduziert die

Algorithmus 1 Ein naiver Algorithmus zur Abstandsberechnung zweier Körper.

Eingabe: Die beiden Körper $\mathcal{K}_1$ und $\mathcal{K}_2$.
Ausgabe: Der Abstand $\delta(\mathcal{K}_1, \mathcal{K}_2)$ und ein nahestes Punktepaar $\chi(\mathcal{K}_1, \mathcal{K}_2)$ der beiden Körper.

```
DISTANCE(K1, K2)
(1)   d* ← ∞
(2)   foreach f1 ∈ F1
(3)     foreach f2 ∈ F2
(4)       [d, (p1, p2)] ← FACEFACEDISTANCE(f1, f2)
(5)       if d < d*
(6)         d* ← d
(7)         p_i* ← p_i   i = 1, 2
(8)   return [d*, (p1*, p2*)]
```

Abstandsberechnung zweier Flächen auf drei elementare Probleme, nämlich die Bestimmung des Abstands zweier Kanten sowie eines Punktes von einer Fläche und den Kante-Fläche-Überlappungstest. Diese elementaren Abstandstests werden in Abschnitt 3.4 diskutiert. Wir wollen an dieser Stelle davon ausgehen, daß wir über eine Routine FACEFACEDISTANCE verfügen, die die Distanz $\delta(f_1, f_2)$ zweier Flächen berechnet und den Abstandswert zusammen mit einem nahesten Punktepaar (p_1, p_2) zurückliefert. Die Vollenumeration der Alternativenmenge $\mathcal{F}_1 \times \mathcal{F}_2$ in Algorithmus 1 ermöglicht die Bestimmung des Abstandsminimums $\delta(\mathcal{K}_1, \mathcal{K}_2) = \min\left\{\delta(f_1, f_2) \,\middle|\, (f_1, f_2) \in \mathcal{F}_1 \times \mathcal{F}_2\right\}$.

Da der Algorithmus selbst in der Praxis eine quadratische Laufzeit in der Flächenzahl besitzt, ist die Verwendbarkeit des naiven Verfahrens im Rahmen einer echtzeitfähigen Kollisionserkennung für Szenarien mit komplexen Körpern stark eingeschränkt. Die Einfachheit der Vorgehensweise erlaubt jedoch ihren Einsatz zur Lösung kleiner Teilprobleme, wie sie beispielsweise von *Divide-and-Conquer-Algorithmen* generiert werden. Gleichzeitig stellt das Verfahren einen Vergleichsmaßstab dar, gegenüber dem die von uns entwickelten Algorithmen ihre Überlegenheit demonstrieren müssen.

3.3.4 Das Branch-and-Bound-Verfahren

In diesem Abschnitt wollen wir darstellen, wie das naive Verfahren aus 3.3.3 mit Hilfe des *Branch-and-Bound-Paradigmas* zur Lösung von Minimierungsproblemen beschleunigt werden kann. Diese Technik erlaubt die praktische Laufzeit durch Reduktion der elementaren Fläche-Fläche-Tests deutlich zu vermindern. Der asymptotische Rechenzeitbedarf für den worst-case erfährt dadurch jedoch keine Verbesserung. Die Branch-and-Bound-Methode zählt auf „intelligente“ Weise die *Alternativenmenge* eines *kombinatorischen Optimierungsproblems* auf. „Intelligent“ bedeutet, daß Teile des Lösungsraums, die als Lieferant des Optimums nicht in Frage kommen, vorzeitig ausgeschlossen werden. Ausgangspunkt ist die endliche Menge X aller *zulässiger Lösungen*, sowie eine

Kostenfunktion $c : X \to \mathbb{R}$. Es spielt dabei keine Rolle, ob c linear ist oder nicht. Die Grundidee ist nun, die Alternativenmenge sukzessive zu partitionieren, die Probleme mit kleinerem zulässigen Bereich zu lösen und über deren Lösung andere Teile des zulässigen Bereiches X auszuschließen. Um ein Branch-and-Bound-Verfahren formulieren zu können, sind lediglich zwei problemspezifische Operationen zu realisieren.

1. *Die Verzweigungsoperation* (*Branching*)
   Die Verzweigungsoperation zerlegt die Alternativenmenge $Y \subseteq X$ in k disjunkte Teilmengen $Y_1, \ldots, Y_k$ mit der Eigenschaft $\cup_{i=1}^{k} Y_i = Y$. Anhand dieser Partitionierungsoperation wird ein *Entscheidungsbaum* aufgebaut, dessen Wurzel das Optimierungsproblem für die vollständige Alternativenmenge X beschreibt und dessen Blätter das Problem auf einzelne zulässige Lösungen $\{x\}$, $x \in X$ beschränken.

2. *Die Berechnung unterer Schranken* (*Lower Bounding*)
   Diese Operation ermittelt für eine Teilmenge Y von X eine untere Schranke μ_Y für den minimalen Kostenfunktionswert $\min\{c(y) \mid y \in Y\}$, der innerhalb der Alternativenmenge Y gefunden werden kann.

Eine Annahme, die man dabei macht, ist, daß die Kosten $c(x)$ einer zulässigen Lösung $x \in X$ „leicht“ berechnet werden können. Der Wert $c(x)$ stellt eine *obere Schranke* für die Kosten $c(x^*)$ der optimalen Lösung x^* dar. Somit liefert jede Lösung eines Optimierungsproblems auf Blattebene eine neue obere Schranke, die zur Aktualisierung einer *globalen oberen Schranke* ν herangezogen werden kann. Der Vergleich einer unteren mit einer oberen Schranke und insbesondere mit der globalen oberen Schranke liefert eine Regel, die die Betrachtung von Teilbäumen des Entscheidungsbaums, d.h. von Teilmengen $Y \subseteq X$ überflüssig macht.

Branch-and-Bound-Ausschlußregel:

Gilt für das partielle Optimierungsproblem , das auf die Alternativenmenge $Y \subseteq X$ beschränkt ist:

$$\mu_Y \geq \nu \,,$$

so kann auf eine Betrachtung der Menge Y verzichtet werden, da gilt:

$$c(x^*) \leq \nu \leq \mu_Y \leq \min\{c(y) \mid y \in Y\} \,.$$

Die rekursive Prozedur FINDMIN (vgl. Algorithmus 2) formuliert das generische Branch-and-Bound-Verfahren als Pseudoalgorithmus. FINDMIN verwendet die Funktion LOWERBOUND zur Berechnung einer unteren Schranke sowie eine Routine PARTITION, die die Menge Y in k disjunkte, nichtleere Teilmengen unterteilt. Zur Bestimmung der optimalen Lösung $[x^*, c(x^*)]$ initialisiert die Routine OPTIMIZE (Algorithmus 3) die globale Variable ν mit ∞ und ruft FINDMIN auf der Alternativenmenge X auf.

Algorithmus 2 Die Branch-and-Bound Rekursion.

Eingabe: Eine Teilmenge Y der Alternativenmenge.

```
FINDMIN(Y)
(1)    (* ν und x* globale Variablen *)
(2)    if Y = {x}
(3)        if c(x) < ν
(4)            ν ← c(x)
(5)            x* ← x
(6)    else
(7)        μ_Y ← LOWERBOUND(Y)
(8)        if ν ≤ μ_Y
(9)            return
(10)       else
(11)           PARTITION(Y, Y_1, ..., Y_k)
(12)           foreach i ∈ {1, ..., k}
(13)               FINDMIN(Y_i)
(14)   return
```

Algorithmus 3 Die Initialisierung des Verfahrens.

Eingabe: Die Alternativenmenge X.
Ausgabe: Die optimale Lösung x^* und dessen Kosten $c(x^*)$.

```
OPTIMIZE(X)
(1)    (* ν und x* globale Variablen *)
(2)    if X = ∅
(3)        return ∅
(4)    ν ← ∞
(5)    x* ← nil
(6)    FINDMIN(X)
(7)    return [x*, c(x*)]
```

Da es sich bei dem vorgestellten Branch-and-Bound-Algorithmus um eine *generische* Beschreibung der algorithmischen Vorgehensweise handelt, müssen wir das Verfahren mit unserer Problemstellung instantiieren. Aufgrund unserer Überlegungen in Abschnitt 3.1 gilt:

1. $X := \mathcal{F}_1 \times \mathcal{F}_2$.
2. $c\big((f_1, f_2)\big) := \delta(f_1, f_2)$.

Die Auswertung der Kostenfunktion für ein einzelnes Flächenpaar $(f_1, f_2) \in X$ entspricht einem elementaren Abstandstest, der in Abschnitt 3.4 erläutert wird. Auf diese

Weise erhalten wir die gesuchten oberen Schranken für den Körperabstand. Weniger offensichtlich ist die Realisierung der zentralen Operationen „Verzweigung“ und „Berechnung unterer Schranken“. Ihre Umsetzung wollen wir an dieser Stelle lediglich skizzieren. Eine detaillierte Darstellung liefern die Abschnitte 3.5 bis 3.8. In 3.5 wollen wir Teile der Oberfläche des Körpers in sogenannte Hüllkörper einschließen. Dabei handelt es sich um kompakte, zusammenhängende, starre Punktmengen, die die entsprechende Flächenmenge einhüllen. Bezeichnet $H(\mathcal{F}_i)$ die Hüllkörper der beiden disjunkten Flächenmengen $\mathcal{F}_i$, $i = 1, 2$, so gilt aufgrund ihrer Hüllkörpereigenschaft:

$$\delta\big(H(\mathcal{F}_1), H(\mathcal{F}_2)\big) \leq \delta(\mathcal{F}_1, \mathcal{F}_2) \,.$$

Somit kann der Hüllkörperabstand zur Berechnungen unterer Schranken im Rahmen des Branch-and-Bound-Verfahrens herangezogen werden. Sei $Y \subseteq X$ und seien $\mathcal{F}_i^Y := \big\{f_i \,\big|\, (f_1, f_2) \in Y\big\}$ die Flächenmengen von Körper $\mathcal{K}_i$, $i = 1, 2$, welche in dem durch Y definierten Teilproblem betrachtet werden. Dann gilt:

$$\delta\big(H(\mathcal{F}_1^Y), H(\mathcal{F}_2^Y)\big) \leq \delta(\mathcal{F}_1^Y, \mathcal{F}_2^Y) = \min\big\{c(x) \,\big|\, x \in Y\big\} \,.$$

Im Zusammenhang mit dem Einsatz von Hüllkörpern müssen die folgenden Fragestellungen untersucht werden:

- Welche *Hüllkörpertypen* können sinnvoll in diesem Kontext eingesetzt werden und wie können diese Typen hinsichtlich ihrer *Eignung* verglichen werden?
- Wie können konkrete Hüllkörper eines bestimmten Typs *konstruiert* werden und welche *Zielsetzungen* sind dabei zu beachten?
- Wie kann auf effiziente Weise der *Abstand* zweier Hüllkörper berechnet werden?

Diese Problemstellungen sind Gegenstand der Abschnitte 3.5 und 3.7.

Bezüglich des anderen Problemkreises, nämlich der Partitionierung der Alternativenmenge sind die folgenden Fragen zu beantworten:

- Soll die Partitionierung *online*, d.h. zur Laufzeit des Branch-and-Bound-Verfahrens erfolgen oder kann sie bereits im Rahmen der *Vorberechnung* sinnvoll festgelegt werden?
- Welche *Zielsetzungen* verfolgen wir mit der Partitionierung und an welchen *Kriterien* können günstige Aufteilungen festgemacht werden?
- In *wie viele Teilmengen* soll die Menge Y unterteilt werden?
- Wie erfolgt die *Verteilung* der Flächenpaare aus Y auf die einzelnen Teilmengen $Y_1, \ldots, Y_k$?

In dem Verfahren, das wir präsentieren werden, werden die Flächenmengen $\mathcal{F}_1$ und $\mathcal{F}_2$ der beiden Körper unabhängig voneinander im Rahmen der Vorberechnung aufgeteilt. Man erhält auf diese Weise eine Hierarchie zunehmend verfeinerter Partitionen von $\mathcal{F}_1$

bzw. $\mathcal{F}_2$. Da wir jede Flächenmenge mit einem Hüllkörper überdecken sprechen wir von einer sogenannten *Hüllkörperhierarchie*. Diese wird in Form eines binären Baumes repräsentiert ($k = 2$). Die erforderliche hierarchische Aufteilung von X wird durch Kombination der Partitionen von $\mathcal{F}_1$ und $\mathcal{F}_2$ zur Laufzeit des Verfahrens konstruiert. Eine umfassende Darstellung dieser Vorgehensweise findet sich in Abschnitt 3.6.

3.4 Elementare Abstandsberechnungen

Im vorangegangenen Abschnitt haben wir einen sehr naiven Algorithmus zur statischen Abstandsberechnung zweier Körper vorgestellt. Dabei wurde der Abstand zwischen den Körpern als minimaler Abstand aller Flächenpaare der beiden Objekte gefunden. Der Einsatz der Branch-and-Bound-Technik soll die Zahl der zu betrachtenden Flächenpaare deutlich reduzieren, indem gewisse Objekteile als zu weit voneinander entfernt klassifiziert werden. Doch auch bei dieser Vorgehensweise ist es erforderlich, für die Flächenpaare, die nicht ausgeschlossen werden können, den exakten Abstand zu ermitteln. Wir wollen uns daher im folgenden mit zwei Fragen beschäftigen:

1. Wie kann der Euklidische Abstand zwischen einem Flächenpaar effizient berechnet werden?

2. Wie kann ein nahestes Punktepaar zweier Flächen bestimmt werden?

Konzeptionell haben wir beide Fragen bereits in Abschnitt 3.1 beantwortet. Dort haben wir festgestellt, daß wir uns bezüglich der Abstandsberechnung auf die Betrachtung der Kantenpaare sowie der Punkt-Fläche-Paare der beiden Flächen zurückziehen können. Allerdings benötigen wir zusätzlich einen Kante-Fläche-Überlappungstest, um im Fall einer Durchdringung der beiden Flächen den korrekten Distanzwert von 0 liefern zu können. Diese elementaren Abstandstests sind zentraler Gegenstand dieses Abschnittes.

3.4.1 Kante-Kante-Abstand

Die Bestimmung des Euklidischen Abstandes zweier Geradensegmente ist ein Problem, das in vielen Bereichen der geometrischen Informationsverarbeitung, sei es in der Bildverarbeitung, im CAD-Umfeld oder im VLSI-Entwurf, auftritt. In [Lum85] wird ein effizienter Algorithmus zur Bestimmung des Abstands zweier Geradensegmente vorgestellt.

Gegeben seien zwei Kanten $e_1 = (a_1, b_1)$ und $e_2 = (a_2, b_2)$, wobei a_i und b_i die Endpunkte von e_i, $i = 1, 2$ darstellen. Der Vektor $\boldsymbol{\nu}_i := \mathbf{b}_i - \mathbf{a}_i$ ist der sogenannte Richtungsvektor der Kante e_i:

$$e_i = \left\{ \mathbf{a}_i + \mu_i \boldsymbol{\nu}_i \,|\, 0 \leq \mu_i \leq 1 \right\} \qquad i = 1, 2\,.$$

Die zugehörigen Geraden bezeichnen wir mit g_1 bzw. g_2:

$$g_i = \left\{ \mathbf{a}_i + \lambda_i \boldsymbol{\nu}_i \,|\, \lambda_i \in \mathbb{R} \right\} \qquad i = 1, 2\,.$$

Somit stellt die Bestimmung des Euklidischen Abstands zweier Geraden im Raum ein relaxiertes Kante-Kante-Abstandsproblem dar. Es gilt stets:

$$\delta(e_1, e_2) \geq \delta(g_1, g_2) \, .$$

Wir wollen zunächst die Fragestellung im Geradenfall untersuchen und ausgehend von den dabei gewonnenen Erkenntnissen ein Verfahren zur Ermittlung des Euklidischen Abstands zweier Kanten vorstellen.

Seien $\mathbf{x}_i = \mathbf{a}_i + \lambda_i \boldsymbol{\nu}_i$, $i = 1, 2$ zwei Geraden g_1 bzw. g_2, dann ergibt sich der quadratische Euklidische Abstand der beiden Punkte als

$$\delta^2(x_1, x_2) = (\mathbf{x}_1 - \mathbf{x}_2)^2 = (\lambda_1 \nu_1 - \lambda_2 \nu_2 - \nu_{12})^2 \, , \tag{3.2}$$

wobei $\boldsymbol{\nu}_{12} := \mathbf{a}_2 - \mathbf{a}_1$ den Verbindungsvektor der beiden Aufpunkte bezeichnet.
Da wir den minimalen Abstand zwischen den Geraden ermitteln möchten, müssen wir 3.2 über λ_1 und λ_2 minimieren:

$$\begin{array}{ll} \min & d(\lambda_1, \lambda_2) = (\lambda_1 \boldsymbol{\nu}_1 - \lambda_2 \boldsymbol{\nu}_2 - \boldsymbol{\nu}_{12})^2 \\ \text{s.d.} & \lambda_1, \lambda_2 \in \mathbb{R} \end{array}$$

Die abstandsminimierenden Parameterwerte bezeichnen wir mit λ_1^* und λ_2^*. Den quadratischen Euklidischen Abstand zwischen den beiden Geraden erhält man nun als Abstand der so gefundenen nahesten Punkte $\chi_i(g_1, g_2) = \mathbf{a}_i + \lambda_i^* \boldsymbol{\nu}_i$, $i = 1, 2$:

$$\delta^2(g_1, g_2) = \delta^2\big(\chi_1(g_1, g_2), \chi_2(g_1, g_2)\big) = (\lambda_1^* \boldsymbol{\nu}_1 - \lambda_2^* \boldsymbol{\nu}_2 - \boldsymbol{\nu}_{12})^2 \, . \tag{3.3}$$

Wir betrachten daher die partiellen Ableitungen bezüglich λ_1 und λ_2:

$$\begin{aligned} \frac{\partial d(\lambda_1, \lambda_2)}{\partial \lambda_1} &= 2 \, (\lambda_1 \boldsymbol{\nu}_1 - \lambda_2 \boldsymbol{\nu}_2 - \boldsymbol{\nu}_{12})^T \boldsymbol{\nu}_1 \, , \\ \frac{\partial d(\lambda_1, \lambda_2)}{\partial \lambda_2} &= -2 \, (\lambda_1 \boldsymbol{\nu}_1 - \lambda_2 \boldsymbol{\nu}_2 - \boldsymbol{\nu}_{12})^T \boldsymbol{\nu}_2 \, . \end{aligned}$$

Die Bedingungen erster Ordnung sind erfüllt, falls

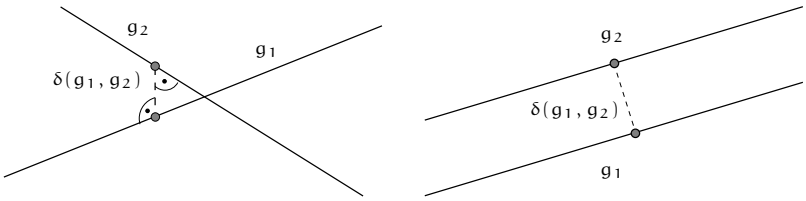
$$\begin{array}{lll} (i) & \lambda_1 \boldsymbol{\nu}_1^2 - \lambda_2 \boldsymbol{\nu}_1^T \boldsymbol{\nu}_2 &= \boldsymbol{\nu}_1^T \boldsymbol{\nu}_{12} \, , \\ (ii) & \lambda_1 \boldsymbol{\nu}_1^T \boldsymbol{\nu}_2 - \lambda_2 \boldsymbol{\nu}_2^2 &= \boldsymbol{\nu}_2^T \boldsymbol{\nu}_{12} \end{array}$$

gilt.
Somit erhalten wir für λ_i^* in Abhängigkeit von dem anderen Parameter:

$$\lambda_1^*(\lambda_2) = \frac{\lambda_2 \boldsymbol{\nu}_1^T \boldsymbol{\nu}_2 + \boldsymbol{\nu}_1^T \boldsymbol{\nu}_{12}}{\boldsymbol{\nu}_1^2} = \frac{\lambda_2 V_{12} + W_1}{V_1} \, , \tag{3.4}$$

$$\lambda_2^*(\lambda_1) = \frac{\lambda_1 \boldsymbol{\nu}_1^T \boldsymbol{\nu}_2 - \boldsymbol{\nu}_2^T \boldsymbol{\nu}_{12}}{\boldsymbol{\nu}_2^2} = \frac{\lambda_1 V_{12} - W_2}{V_2} \, , \tag{3.5}$$

Abbildung 3.7 Der minimale Abstand zwischen g_1 und g_2.



allgemeiner Fall parallele Geraden

wobei $V_{1,2}$, V_i, und W_i, $i = 1, 2$ folgendermaßen definiert sind:

$$V_{12} := v_1^T v_2 \qquad V_i := v_i^2, \qquad W_i := v_i^T v_{12}, \quad i = 1, 2\,. \tag{3.6}$$

Einsetzen liefert eine explizite Darstellung von λ_i^*, $i = 1, 2$:

$$\lambda_1^* = \frac{\boldsymbol{v}_1^T \boldsymbol{v}_{12} \boldsymbol{v}_2^2 - \boldsymbol{v}_2^T \boldsymbol{v}_{12} \boldsymbol{v}_1^T \boldsymbol{v}_2}{\boldsymbol{v}_1^2 \boldsymbol{v}_2^2 - (\boldsymbol{v}_1^T \boldsymbol{v}_2)^2} = \frac{W_1 V_2 - W_2 V_{12}}{V_1 V_2 - V_{12}^2}\,, \tag{3.7}$$

$$\lambda_2^* = \frac{\boldsymbol{v}_1^T \boldsymbol{v}_{12} \boldsymbol{v}_1^T \boldsymbol{v}_2 - \boldsymbol{v}_2^T \boldsymbol{v}_{12} \boldsymbol{v}_1^2}{\boldsymbol{v}_1^2 \boldsymbol{v}_2^2 - (\boldsymbol{v}_1^T \boldsymbol{v}_2)^2} = \frac{W_1 V_{12} - W_2 V_1}{V_1 V_2 - V_{12}^2}\,, \tag{3.8}$$

falls $V_1 V_2 - V_{12}^2 = \boldsymbol{v}_1^2 \boldsymbol{v}_2^2 - (\boldsymbol{v}_1^T \boldsymbol{v}_2)^2 = (\boldsymbol{v}_1 \times \boldsymbol{v}_2)^2 \neq 0$ gilt.

Um die soeben gewonnen Ergebnisse besser verstehen zu können wollen wir diese zunächst geometrisch interpretieren. Die Parameter λ_1^* und λ_2^* sind nicht definiert, falls der Nenner von 3.7 und 3.8 Null wird. Offensichtlich ist dies genau dann der Fall, wenn die beiden Geraden parallel sind (unter der Annahme $\boldsymbol{v}_i \neq 0$, $i = 1, 2$). In diesem Fall liegt eine entartete Situation vor, die als zweidimensionales Problem aufgefaßt werden kann (Abbildung 3.7). Als naheste Punkte können wir den Aufpunkt a_1 von g_1 sowie den Punkt auf g_2 mit minimalem Abstand zu a_1 wählen. Letzterer wird durch den Parameter $\lambda_2^*(0)$ beschrieben. Wir erhalten somit:

$$\begin{aligned} \chi(g_1, g_2) &:= \big(a_1, \lambda_2^*(0)\big)\,, \\ \delta^2(g_1, g_2) &= d\big(0, \lambda_2^*(0)\big) = \boldsymbol{v}_{12}^2 - \frac{(\boldsymbol{v}_2^T \boldsymbol{v}_{12})^2}{\boldsymbol{v}_2^2} = \boldsymbol{v}_{12}^2 - \frac{W_2^2}{V_2}\,. \end{aligned}$$

Im allgemeinen Fall, der in Abbildung 3.7 dargestellt ist, wollen wir nun zeigen, daß der minimale Abstand der beiden Geraden der Länge des auf g_1 und g_2 senkrecht stehenden Verbindungssegments s entspricht. Der normierte Richtungsvektor von s ergibt sich somit als $\hat{\boldsymbol{s}} = \frac{\boldsymbol{v}_1 \times \boldsymbol{v}_2}{\|\boldsymbol{v}_1 \times \boldsymbol{v}_2\|}$. Wir suchen also nach der Lösung des folgenden Gleichungssystems:

$$\begin{aligned} & \boldsymbol{a}_1 + \lambda_1 \boldsymbol{v}_1 + d\hat{\boldsymbol{s}} = \boldsymbol{a}_2 + \lambda_2 \boldsymbol{v}_2 \\ \iff \qquad & \big[\, \boldsymbol{v}_1\,, -\boldsymbol{v}_2\,, \hat{\boldsymbol{s}} \,\big] \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ d \end{bmatrix} = \boldsymbol{v}_{12}\,. \end{aligned}$$

Da $g_1 \nparallel g_2$ und $\boldsymbol{v}_i \perp \hat{\boldsymbol{s}}$ für $i = 1, 2$ gilt, folgt aus der linearen Unabhängigkeit von v_1, v_2 und $\hat{s}$: $\det(\boldsymbol{v}_1, \boldsymbol{v}_2, \hat{\boldsymbol{s}}) \neq 0$. Das lineare Gleichungssystem hat somit die folgende Lösung:

$$\begin{aligned}
\lambda_1^* &= \frac{\det(\boldsymbol{v}_{12}, -\boldsymbol{v}_2, \hat{\boldsymbol{s}})}{\det(\boldsymbol{v}_1, -\boldsymbol{v}_2, \hat{\boldsymbol{s}})} = \frac{\boldsymbol{v}_1^T \boldsymbol{v}_{12} \boldsymbol{v}_2^2 - \boldsymbol{v}_2^T \boldsymbol{v}_{12} \boldsymbol{v}_1^T \boldsymbol{v}_2}{\boldsymbol{v}_1^2 \boldsymbol{v}_2^2 - (\boldsymbol{v}_1^T \boldsymbol{v}_2)^2}\,, \\
\lambda_2^* &= \frac{\det(\boldsymbol{v}_1, \boldsymbol{v}_{12}, \hat{\boldsymbol{s}})}{\det(\boldsymbol{v}_1, -\boldsymbol{v}_2, \hat{\boldsymbol{s}})} = \frac{\boldsymbol{v}_1^T \boldsymbol{v}_{12} \boldsymbol{v}_1^T \boldsymbol{v}_2 - \boldsymbol{v}_2^T \boldsymbol{v}_{12} \boldsymbol{v}_1^2}{\boldsymbol{v}_1^2 \boldsymbol{v}_2^2 - (\boldsymbol{v}_1^T \boldsymbol{v}_2)^2}\,, \\
d^* &= \frac{\det(\boldsymbol{v}_1, -\boldsymbol{v}_2, \boldsymbol{v}_{12})}{\det(\boldsymbol{v}_1, -\boldsymbol{v}_2, \hat{\boldsymbol{s}})} = \sqrt{\left(\lambda_1^* \boldsymbol{v}_1 - \lambda_2^* \boldsymbol{v}_2 - \boldsymbol{v}_{12}\right)^2}\,.
\end{aligned}$$

Die nahesten Punkte zweier Geraden g_1 und g_2 ergeben sich durch folgende Zuordnung:

$$\chi(g_1, g_2) := \begin{cases} \left(g_1(\lambda_1^*), g_2(\lambda_2^*)\right) & \text{falls } g_1 \nparallel g_2\,; \\ \left(a_1, \lambda_2^*(0)\right) & \text{sonst}\,. \end{cases}$$

Dabei bezeichnet $g_i(\lambda_i)$, $\lambda_i \in \mathbb{R}$, den Punkt auf der Geraden g_i mit Ortsvektor $\boldsymbol{a}_i + \lambda_i \boldsymbol{v}_i$, $i = 1, 2$.

Der quadratische Euklidische Abstand zwischen den beiden Geraden kann folgendermaßen ermittelt werden:

$$\begin{aligned}
\delta^2(g_1, g_2) &= \delta^2\left(\chi_1(g_1, g_2), \chi_2(g_1, g_2)\right) \\
&= \begin{cases} \left(\lambda_1^* \boldsymbol{v}_1 - \lambda_2^* \boldsymbol{v}_2 - \boldsymbol{v}_{12}\right)^2 & \text{falls } g_1 \nparallel g_2\,; \\ \boldsymbol{v}_{12}^2 - \frac{(\boldsymbol{v}_2^T \boldsymbol{v}_{12})^2}{\boldsymbol{v}_2^2} & \text{sonst}\,. \end{cases}
\end{aligned}$$

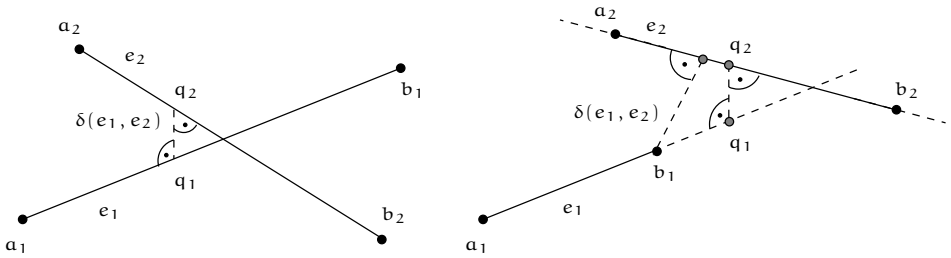
Die endliche Länge von Kanten erschwert die Berechnung ihres Euklidischen Abstands. Während λ_1^* und λ_2^* die nahesten Punkte der mit e_1 und e_2 assoziierten Geraden g_1 und g_2 charakterisieren, beschreiben wir die Punkte minimalen Kantenabstands mit den Parametern μ_1^* bzw. μ_2^*. Wir wollen im folgenden zeigen, wie μ_1^* und μ_2^* in Abhängigkeit von den Werten für λ_1^* und λ_2^* bestimmt werden können. Analog zum Geradenfall erhält man den quadratischen Euklidischen Abstand zwischen den beiden Kanten als Abstand der nahesten Punkte $\chi_i(e_1, e_2) = e_i(\mu_i^*)$, mit $e_i(\mu_i^*) := \boldsymbol{a}_i + \mu_i^* \boldsymbol{v}_i$, $i = 1, 2$.

$$\delta^2(e_1, e_2) = \delta^2\left(\chi_1(e_1, e_2), \chi_2(e_1, e_2)\right) = \left(\mu_1^* \boldsymbol{v}_1 - \mu_2^* \boldsymbol{v}_2 - \boldsymbol{v}_{12}\right)^2\,. \tag{3.9}$$

Ist ein durch den Parameterwert μ_1 beschriebener Punkt von e_1 gegeben, so erhält man den Punkt mit minimalem Abstand auf der mit e_2 assoziierten Geraden g_2, indem man μ_1 in 3.5 einsetzt. Der Parameterwert $\lambda_2^*(\mu_1)$ charakterisiert den Fußpunkt des Lotes auf g_2.

Wir werden häufig den Punkt einer Kante bestimmen müssen, der bezüglich seiner Parametrisierung möglichst nahe zu einem gegebenen Punkt der entsprechenden

Abbildung 3.8 Der minimale Abstand zwischen e_1 und e_2.



Geraden liegt. Hierzu definieren wir die folgende Funktion:

$$\begin{aligned} \tau : \mathbb{R} &\longrightarrow [0,1] \subseteq \mathbb{R}^+ \\ \tau(\lambda) &\longmapsto \begin{cases} 0 & \text{falls } \lambda < 0\,; \\ 1 & \text{falls } \lambda > 1\,; \\ \lambda & \text{sonst}\,. \end{cases} \end{aligned}$$

Ist $\mathbf{x} = \mathbf{a} + \lambda\mathbf{v}$ ein Punkt der Geraden g, so ergibt sich der zu $\mathbf{x}$ am nahesten gelegene Punkt der Kante e als $\tilde{\mathbf{x}} = \mathbf{a} + \tilde{\lambda}\mathbf{v}$, mit $\tilde{\lambda} := \tau(\lambda)$.

Wir wollen im folgenden drei Fälle unterscheiden, die sich an den Werten von λ_1^* und λ_2^* orientieren. Dabei bezeichnen wir die nahesten Punkte zweier nichtparalleler Geraden mit q_1 und q_2.

1.Fall: $\lambda_1^* \in [0,1] \quad \wedge \quad \lambda_2^* \in [0,1]$

Der minimale Abstand zwischen g_1 und g_2 wird an Punkten auf den Kanten e_1 und e_2 angenommen. Abbildung 3.8 veranschaulicht die Situation. Es macht somit keinen Unterschied, ob man zur Abstandsberechnung die Kanten oder die entsprechenden Geraden betrachtet, da die nahesten Punkte von g_1 und g_2 auch Punkte minimalen Kantenabstandes darstellen. Somit gilt:

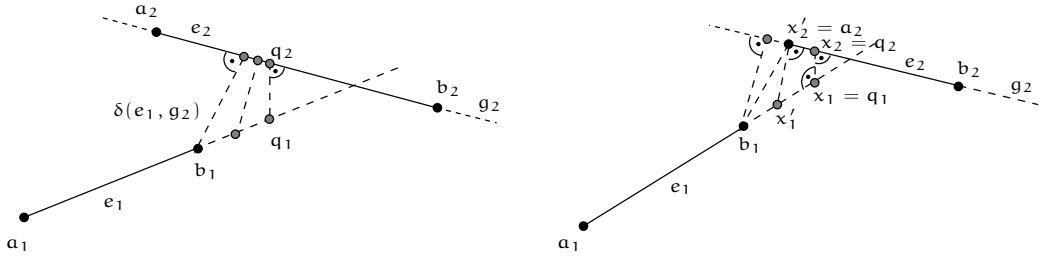
$$\mu_1^* = \lambda_1^* \qquad \text{und} \qquad \mu_2^* = \lambda_2^*\,.$$

Der minimale Abstand zwischen den beiden Kanten kann unmittelbar mit Hilfe von 3.9 bestimmt werden.

2.Fall: $\lambda_1^* \notin [0,1] \quad \vee \quad \lambda_2^* \notin [0,1]$

Wir betrachten nun den Fall, daß mindestens ein nahester Punkt des Geradenpaares (g_1, g_2) außerhalb der mit ihm assoziierten Kante liegt. Eine solche Situation ist im rechten Teil von Abbildung 3.8 dargestellt. Das folgende Lemma wird uns bei der Bestimmung der nahesten Punkte hilfreich sein.

Abbildung 3.9 Die beiden Fälle in dem Beweis von Lemma 3.4.



$x_1 = b_1$ bevor $x_2 = a_2$ $\quad$ $x_2 = a_2$ bevor $x_1 = b_1$

Lemma 3.3. *Gilt für eine durch λ_1^* parametrisierte Gerade g_1, daß λ_1^* außerhalb des Intervalls $[0,1]$ liegt, dann entspricht der minimale Abstand zwischen der Kante e_1 und der Geraden g_2 dem minimalen Abstand eines Endpunktes von e_1 zu g_2. Dabei handelt es sich um den Endpunkt von e_1, der bezüglich seiner Parametrisierung am nahesten zu λ_1^* liegt.*

Beweis. Sei $\lambda_1^* > 1$. Dies bedeutet, daß q_1 näher zu b_1 als zu a_1 liegt. Da g_1 und e_1 geradlinig sind, nimmt der Abstand monoton zu, wenn wir einen Punkt x_1 ausgehend von q_1 auf g_1 in Richtung von e_1 bewegen. Der linke Teil von Abbildung 3.9 zeigt die Verschiebung. Der Endpunkt b_1 ist der erste Punkt von e_1, der bei der Bewegung von x_1 erreicht wird. Es muß sich somit um den nahesten Punkt von e_1 zu g_2 handeln.
Falls $\lambda_1^* < 0$ gilt, so liefert eine analoge Argumentation die Erkenntnis, daß a_1 den minimalen Abstand von e_1 zu g_2 definiert. □

Wir können nun eine Aussage über das naheste Punktepaar machen, falls *genau ein* λ-Parameter außerhalb des Intervalls $[0,1]$ liegt.

Lemma 3.4. *Gilt für die durch λ_i^* parametrisierten Geraden g_i, $i = 1,2$, daß λ_1^* außerhalb und λ_2^* innerhalb des Intervalls $[0,1]$ liegt, so wird der minimale Abstand zwischen e_1 und e_2 an einem der Endpunkte von e_1 angenommen. Desweiteren handelt es sich dabei um den Endpunkt von e_1, der bezüglich seiner Parametrisierung am nahesten zu λ_1^* liegt.*

Beweis. Sei $\lambda_1^* > 1$ und $\lambda_2^* \in [0,1]$. Anhand der Parametrisierung von g_1 können wir schließen, daß b_1 näher zu q_1 liegt als a_1. Bewegen wir uns ausgehend von q_1 auf der Geraden g_1 in Richtung von e_1, so finden wir zu jedem Punkt x_1 auf e_1 einen Punkt x_2 auf g_2, an dem der minimale Abstand zwischen x_1 und e_2 angenommen wird. Den Punkt x_2 erhalten wir, indem wir das Lot von x_1 auf g_2 fällen. Während sich x_1 auf der Geraden g_1 in Richtung von b_1 bewegt, nähert sich x_2 einem Endpunkt von e_2 – in Abbildung 3.9 dem Punkt a_2. Da g_1 und g_2 Geraden sind wird der Abstand zwischen x_1 und g_2 monoton wachsen.
Wir können nun zwei Fälle unterscheiden:

1. Falls x_1 den Endpunkt b_1 erreicht, bevor x_2 mit dem Endpunkt a_2 zusammenfällt, dann sind $x_1 = b_1$ und x_2 das Punktepaar, das den minimalen Abstand zwischen e_1 und e_2 definiert.

2. Im anderen Fall muß der von x_2 erreichte Endpunkt auch bei weiterer Verschiebung von x_1 der naheste Punkt zu x_1 sein. Das Lot von x_1 auf g_2 liefert nämlich einen Punkt, der bezüglich seiner Parametrisierung näher zu diesem Endpunkt als zu dem anderen liegt, wie der rechte Teil von Abbildung 3.9 verdeutlicht. Somit entspricht nach Lemma 3.3 der naheste Punkt von e_2 zu g_1 dem von x_2 erreichten Endpunkt von e_2. Sein Pendant auf e_1 kann nur b_1 sein, da dies der erste Punkt ist, welcher durch weitere Verschiebung von x_1 erreicht wird.

In beiden Fällen ist b_1 der naheste Punkt zwischen den beiden Kanten auf e_1.

Falls $\lambda_1^* < 0$ und $\lambda_2^* \in [0,1]$ gilt, ergibt sich aufgrund einer analogen Beobachtung, daß a_1 der naheste Punkt auf e_1 ist. □

Lemma 3.4 gestattet, einen Endpunkt einer Kante als nahesten Punkt zu wählen, sobald der abstandsminimierende Parameter der entsprechenden Geraden außerhalb und der der anderen Geraden innerhalb des Intervalls $[0,1]$ liegt. Es bleiben jedoch zwei Fragen offen:

1. Wie wird der naheste Punkt der Kante bestimmt, dessen λ^*-Parameter in das Intervall $[0,1]$ fällt?

2. Wie können die abstandsminimierenden Punkte der beiden Kanten bestimmt werden, falls *beide* Parameter außerhalb des Intervalls liegen?

Frage 1 haben wir prinzipiell im Beweis zu Lemma 3.4 beantwortet. Dabei wurde der Punkt x_1 ausgehend von q_1 auf g_1 in Richtung von e_1 verschoben. Der naheste Punkt auf e_2 ergab sich als Fußpunkt x_2 des Lotes von b_1 ($\lambda_1^* > 1$) bzw. a_1 ($\lambda_1^* < 0$) auf e_2 (Fall 1) oder als der x_2 am nahesten gelegene Endpunkt von e_2, falls der x_2 charakterisierende Parameter außerhalb des Intervalls $[0,1]$ lag (Fall 2).

Mit Hilfe von 3.5 können wir den Lotfußpunkt des durch Lemma 3.4 bestimmten Endpunkts von e_1 auf g_2 bestimmen. Befindet sich der so gefundene Punkt auf der Kante, so haben wir den nahesten Punkt auf e_2 gefunden, andernfalls wählen wir den Endpunkt von e_2, der bezüglich der λ^*-Parametrisierung am nahesten zu dem Lotfußpunkt liegt. Somit erhalten wir:

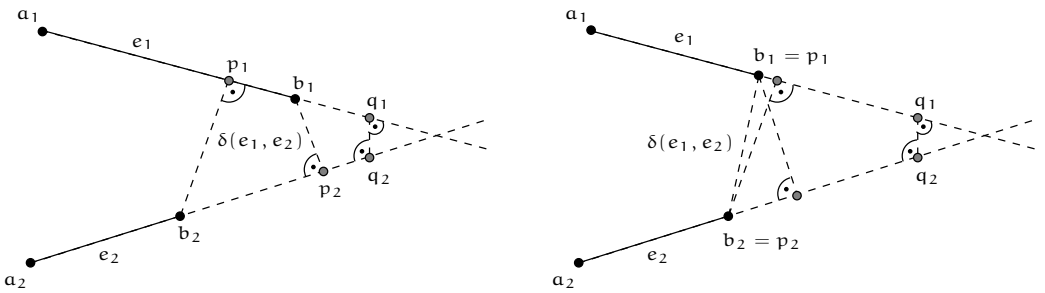
$$\mu_1^* = \begin{cases} \tau(\lambda_1^*) & \text{falls } \lambda_1^* \notin [0,1] \wedge \lambda_2^* \in [0,1]\,; \\ \tau \circ \lambda_1^* \circ \tau(\lambda_2^*) & \text{falls } \lambda_1^* \in [0,1] \wedge \lambda_2^* \notin [0,1]\,. \end{cases}$$

$$\mu_2^* = \begin{cases} \tau \circ \lambda_2^* \circ \tau(\lambda_1^*) & \text{falls } \lambda_1^* \notin [0,1] \wedge \lambda_2^* \in [0,1]\,; \\ \tau(\lambda_2^*) & \text{falls } \lambda_1^* \in [0,1] \wedge \lambda_2^* \notin [0,1]\,. \end{cases}$$

Dabei bezeichnet $\circ$ die Hintereinanderausführung der Funktionen τ und λ_i^*.

Sind die Voraussetzungen von Lemma 3.4 nicht erfüllt, so müssen wir eine Antwort auf Frage 2 finden, wollen wir den Abstand zwischen den beiden Kanten ermitteln. Liegen beide Parameter λ_1^* und λ_2^* außerhalb des Kantenintervalls , so kann Lemma 3.4 nicht unmittelbar zur Bestimmung der nahesten Punkte herangezogen werden. Die Bedingung $\lambda_i^* \notin [0,1]$ für $i = 1,2$ ist nicht hinreichend, beide Endpunkte als naheste

Abbildung 3.10 Der minimale Abstand zwischen e_1 und e_2: Fall 2b .



Punkte der beiden Kanten zu wählen, wie Abbildung 3.10 zeigt. In Beispiel (a) von Abbildung 3.10 ist b_1 zwar der naheste Punkt zwischen e_1 und g_2, doch p_1 ist der Punkt auf e_1, an dem der minimale Abstand zwischen e_1 und e_2 angenommen wird. Der Grund ist, daß der Lotfußpunkt von b_1 auf g_2 außerhalb der Kante e_2, der von b_2 auf g_1 jedoch innerhalb der Kante e_1 liegt. Befinden sich allerdings beide Fußpunkte außerhalb der entsprechenden Kanten, wie in Beispiel (b) von Abbildung 3.10 zu sehen, so wird der minimale Abstand zwischen den Endpunkten von e_1 und e_2 angenommen, deren Parameter am nahesten zu λ_1^* und λ_2^* liegen.

Es ist nun leicht einzusehen, daß beide Fälle durch höchstens zweimalige Anwendung von Lemma 3.4 behandelt werden können. Betrachten wir wieder Beispiel (a). Wir wählen zunächst $\mu_1^* := 1$, da $\lambda_1^* > 1$ ist, wenden anschließend Lemma 3.4 an und erhalten aufgrund von $\lambda_2^*(\mu_1^*) > 1$ den nahesten Punkt auf e_2 durch $\mu_2^* := 1$. Eine weitere Anwendung von Lemma 3.4 liefert mit $\lambda_1^*(\mu_2^*) \in [0,1]$ das Pendant auf e_1. In Beispiel (b) würde sich dagegen nach der zweiten Anwendung $\lambda_1^*(\mu_2^*) > 1$ ergeben, weshalb wir μ_1^* auf den korrekten Wert 1 setzen würden. In beiden Fällen hat also die zweimalige Anwendung von Lemma 3.4 das naheste Punktepaar geliefert. Für den Unterfall, daß *beide* Parameter außerhalb des Kantenintervalls liegen, ergibt sich somit:

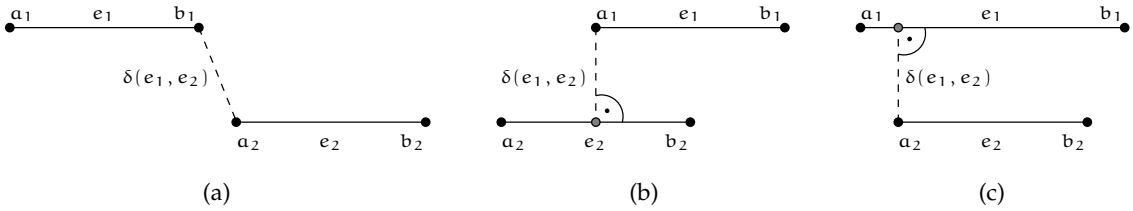
$$\mu_1^* = \begin{cases} \tau(\lambda_1^*) & \text{falls } \lambda_2^* \circ \tau(\lambda_1^*) \in [0,1]\,; \\ \tau \circ \lambda_1^* \circ \tau \circ \lambda_2^* \circ \tau(\lambda_1^*) & \text{falls } \lambda_2^* \circ \tau(\lambda_1^*) \notin [0,1]\,. \end{cases}$$

$$\mu_2^* = \begin{cases} \lambda_2^* \circ \tau(\lambda_1^*) & \text{falls } \lambda_2^* \circ \tau(\lambda_1^*) \in [0,1]\,; \\ \tau \circ \lambda_2^* \circ \tau(\lambda_1^*) & \text{falls } \lambda_2^* \circ \tau(\lambda_1^*) \notin [0,1]\,. \end{cases}$$

3.Fall: λ_1^*, λ_2^* nicht definiert

Betrachten wir Kantenpaare, so kann der Nenner von 3.7 bzw. 3.8 Null werden, wenn eine der beiden Kanten zu einem Punkt entartet oder die beiden Kanten parallel sind. Ersteres wollen wir für unsere Randrepräsentation nicht erlauben, so daß wir davon ausgehen können, daß die beiden Kanten die gleiche Richtung $\boldsymbol{\nu}$ haben. Unsere Ab-

Abbildung 3.11 Der minimale Abstand zwischen e_1 und e_2: Fall 3.



standsberechnungsproblem hat somit nur noch zweidimensionalen Charakter.

Wir nehmen im folgenden an, daß $\mathbf{a}_1^T\boldsymbol{\nu} < \mathbf{b}_1^T\boldsymbol{\nu}$ und $\mathbf{a}_2^T\boldsymbol{\nu} < \mathbf{b}_2^T\boldsymbol{\nu}$, was durch Auswertung von vier Skalarprodukten und zwei Vergleichen überprüft werden kann. Sollte eines der Prädikate nicht erfüllt sein, so ergibt sich die folgende Fallunterscheidung durch Umbenennung. In Bezug auf die Lage des nahesten Punktepaares genügt es zwei Fälle zu unterscheiden.

1. $b_1^T\nu \leq a_2^T\nu \quad \vee \quad b_2^T\nu \leq a_1^T\nu$
   Das naheste Punktepaar wird von den Endpunkten der beiden Kanten gebildet. Die Situation ist in Abbildung 3.11 (a) dargestellt. Der minimale Abstand zwischen e_1 und e_2 ergibt sich als Euklidischer Abstand der beiden Endpunkte b_1 und a_2 bzw. a_1 und b_2

$$\mu_1^* = \begin{cases} 0 & \text{falls } \mathbf{b}_2^T\boldsymbol{\nu} > \mathbf{a}_1^T\boldsymbol{\nu}\,; \\ 1 & \text{falls } \mathbf{b}_1^T\boldsymbol{\nu} \leq \mathbf{a}_2^T\boldsymbol{\nu}\,. \end{cases}$$

$$\mu_2^* = \begin{cases} 1 & \text{falls } \mathbf{b}_2^T\boldsymbol{\nu} > \mathbf{a}_1^T\boldsymbol{\nu}\,; \\ 0 & \text{falls } \mathbf{b}_1^T\boldsymbol{\nu} \leq \mathbf{a}_2^T\boldsymbol{\nu}\,. \end{cases}$$

2. $\mathbf{b}_1^T\boldsymbol{\nu} > \mathbf{a}_2^T\boldsymbol{\nu} \quad \wedge \quad \mathbf{b}_2^T\boldsymbol{\nu} > \mathbf{a}_1^T\boldsymbol{\nu}$
   Das naheste Punktepaar entspricht einem Endpunkt einer Kante und einem inneren Punkt der anderen Kante. Anhand von Abbildung 3.11 (b),(c) sieht man leicht ein, daß lediglich ein weiterer Vergleich benötigt wird, um zu entscheiden, welche Kante den Endpunkt und welche den inneren Punkt des nahesten Punktepaares stellt. Gilt $\mathbf{a}_1^T\boldsymbol{\nu} \leq \mathbf{a}_2^T\boldsymbol{\nu}$, so ermitteln wir den minimalen Abstand von a_2 zu e_1, andernfalls von a_1 zu e_2.

$$\mu_1^* = \begin{cases} 0 & \text{falls } \mathbf{a}_1^T\boldsymbol{\nu} > \mathbf{a}_2^T\boldsymbol{\nu}\,; \\ \lambda_1^*(0) & \text{falls } \mathbf{a}_1^T\boldsymbol{\nu} \leq \mathbf{a}_2^T\boldsymbol{\nu}\,. \end{cases}$$

$$\mu_2^* = \begin{cases} \lambda_2^*(0) & \text{falls } \mathbf{a}_1^T\boldsymbol{\nu} > \mathbf{a}_2^T\boldsymbol{\nu}\,; \\ 0 & \text{falls } \mathbf{a}_1^T\boldsymbol{\nu} \leq \mathbf{a}_2^T\boldsymbol{\nu}\,. \end{cases}$$

Eine andere Möglichkeit den Abstand zweier paralleler Kanten e_1 und e_2 zu ermitteln und die zudem besser zu unserer bisherigen Vorgehensweise paßt, verwendet die

Ergebnisse aus Fall 2, wenn beide λ^* Parameter außerhalb des Kantenintervalls liegen. Haben beide Kanten die gleiche Richtung, so wird der minimale Abstand entweder an zwei Endpunkten (Abbildung 3.11 (a)) oder am Endpunkt einer Kante und dem Fußpunkt des Lotes auf die andere Kante (Abbildung 3.11 (b),(c)) angenommen. Somit gilt:

$$\mu_1^* = \begin{cases} 0 & \text{falls } \lambda_2^*(0) \in [0,1]\,; \\ \tau \circ \lambda_1^* \circ \tau \circ \lambda_2^*(0) & \text{falls } \lambda_2^*(0) \notin [0,1]\,. \end{cases}$$

$$\mu_2^* = \begin{cases} \lambda_2^*(0) & \text{falls } \lambda_2^*(0) \in [0,1]\,; \\ \tau \circ \lambda_2^*(0) & \text{falls } \lambda_2^*(0) \notin [0,1]\,. \end{cases}$$

Algorithmus 4 faßt unsere Überlegungen zu den drei vorgestellten Fällen des Kante-Kante-Abstandsproblems zusammen.

Algorithmus 4 Ein Algorithmus zur Bestimmung des Kante-Kante-Abstands.

Eingabe: Zwei Kanten $e_1 = (a_1, b_1)$ und $e_2 = (a_2, b_2)$ im $\mathbb{R}_3$.
Ausgabe: Der quadratische Euklidische Abstand der beiden Kanten sowie zwei naheste Punkte als Zeugen des Abstandsminimums.

EDGEEDGEDISTANCE(e_1, e_2)
(1) Berechne V_1, V_2, V_{12}, W_1 und W_2 anhand von 3.6
(2) **if** $V_1V_2 - V_{12}^2 = 0$
(3) $\quad \mu_1^* \leftarrow 0$
(4) **else**
(5) $\quad$ Berechne λ_1^* anhand von 3.7.
(6) $\quad \mu_1^* \leftarrow \tau(\lambda_1^*)$
(7) $\mu_2^* \leftarrow \lambda_2^*(\mu_1^*)$
(8) **if** $\mu_2^* \in [0,1]$
(9) $\quad$ **return** $\left[d(\mu_1^*, \mu_2^*), (e_1(\mu_1^*), e_2(\mu_2^*))\right]$
(10) **else**
(11) $\quad \mu_2^* \leftarrow \tau(\mu_2^*)$
(12) $\quad \mu_1^* \leftarrow \lambda_1^*(\mu_2^*)$
(13) $\quad$ **if** $\mu_1^* \in [0,1]$
(14) $\quad\quad$ **return** $\left[d(\mu_1^*, \mu_2^*), (e_1(\mu_1^*), e_2(\mu_2^*))\right]$
(15) $\quad$ **else**
(16) $\quad\quad \mu_1^* \leftarrow \tau(\mu_1^*)$
(17) $\quad\quad$ **return** $\left[d(\mu_1^*, \mu_2^*), (e_1(\mu_1^*), e_2(\mu_2^*))\right]$

3.4.2 Der Punkt-Kante-Abstand

Die Überlegungen des vorangegangenen Abschnitts zur Abstandsberechnung zweier Kanten haben implizit eine weitere Problemstellung im Zusammenhang mit der Abstandsberechnung zwischen Oberflächenelementen gelöst. Die Abstandsberechnung

zwischen einem Eckpunkt und einer Kante kann als Spezialfall des Kante-Kante-Tests aufgefaßt werden, bei dem eine Kante zu einem Punkt entartet ist.

Sei a_2 ein Eckpunkt und $e_1 = \{\mathbf{a}_1 + \mu_1\mathbf{v}_1 \,|\, 0 \leq \mu_1 \leq 1\}$ eine Kante, deren Abstand zu a_2 bestimmt werden soll. Dann lautet der Parameterwert λ_1^* des nahesten Punktes zu a_2 auf der mit e_1 assoziierten Geraden $g_1 = \{\mathbf{a}_1 + \lambda_i\mathbf{v}_1 \,|\, \lambda_1 \in \mathbb{R}\}$:

$$\lambda_1^* = \frac{\mathbf{v}_1^T\mathbf{v}_{12}}{\mathbf{v}_1^2} = \frac{W_1}{V_1}\,,$$

wobei $\mathbf{v}_{12} = a_2 - a_1$ den Verbindungsvektor der Aufpunkte bezeichnet.
Diese Gleichung ergibt sich unmittelbar aus 3.4, indem wir λ_2 auf Null setzen. Anschaulich bezeichnet λ_1^* den Fußpunkt des Lotes durch a_2 auf die Gerade g_1. Wir können nun zwei Fälle unterscheiden:

1. Fall: $\lambda_1^* \in [0,1]$

Falls der naheste Punkt auf g_1 innerhalb des Kantenintervalls liegt, so charakterisiert $\mu_1^* := \lambda_1^*$ den nahesten Punkt zu a_2 auf e_1. Als nahestes Punktepaar erhalten wir:

$$\chi(e_1, a_2) := \big(e_1(\mu_1^*), a_2\big)\,.$$

Der quadratische Euklidische Abstand zwischen e_1 und a_2 beträgt nach 3.9:

$$\begin{aligned}
\delta^2(e_1, a_2) &= (\mu_1^*\mathbf{v}_1 - \mathbf{v}_{12})^2 \\
&= {\mu_1^*}^2\mathbf{v}_1^2 - 2\mu_1^*\mathbf{v}_1^T\mathbf{v}_{12} + \mathbf{v}_{12}^2 \\
&= \frac{W_1^2}{V_1^2}V_1 - 2\frac{W_1}{V_1}W_1 + \mathbf{v}_{12}^2 \\
&= \mathbf{v}_{12}^2 - \frac{W_1^2}{V_1}\,.
\end{aligned}$$

2. Fall: $\lambda_1^* \notin [0,1]$

Befindet sich der durch λ_1^* beschriebene Punkt außerhalb von e_1 so können wir Lemma 3.3 anwenden und erhalten als nahesten Punkt von e_1 zu a_2 den Endpunkt von e_1, der bezüglich seiner Parametrisierung am nahesten zu λ_1^* liegt:

$$\begin{aligned}
\mu_1^* &:= \begin{cases} 0 & \text{falls } \lambda_1^* < 0\,; \\ 1 & \text{falls } \lambda_1^* > 1\,. \end{cases} \\
\chi(e_1, a_2) := \big(e_1(\mu_1^*), a_2\big)) &= \begin{cases} a_1 & \text{falls } \lambda_1^* < 0\,; \\ b_1 & \text{falls } \lambda_1^* > 1\,. \end{cases}
\end{aligned}$$

Der quadratische Abstand zwischen e_1 und a_2 ergibt sich somit als:

$$\delta^2(e_1, a_2) = \begin{cases} \mathbf{v}_{12}^2 = (\mathbf{a}_2 - \mathbf{a}_1)^2 & \text{falls } \lambda_1^* < 0\,; \\ (\mathbf{a}_2 - \mathbf{b}_1)^2 & \text{falls } \lambda_1^* > 1\,. \end{cases}$$

Zusammenfassend gilt also:

$$\begin{aligned}\chi(e_1, a_2) &:= \big(e_1(\tau(\lambda_1^*)), a_2\big)\\ \delta^2(e_1, a_2) &= \begin{cases} \boldsymbol{v}_{12}^2 & \text{falls } \lambda_1^* < 0\,;\\ (\boldsymbol{a}_2 - \boldsymbol{b}_1)^2 & \text{falls } \lambda_1^* > 1\,;\\ \boldsymbol{v}_{12}^2 - \frac{W_1^2}{V_1} & \text{sonst}\,. \end{cases}\end{aligned}$$

Wir erhalten somit das in Algorithmus 5 beschriebene Verfahren zur Bestimmung des Punkt-Kante-Abstands.

Algorithmus 5 Ein Algorithmus zur Bestimmung des Punkt-Kante-Abstands.

Eingabe: Eine Kante $e_1 = (a_1, b_1)$ und ein Eckpunkt a_2 im $\mathbb{R}^3$.
Ausgabe: Der quadratische Euklidische Abstand zwischen der Kante und dem Eckpunkt sowie die beiden nahesten Punkte als Zeugen des Abstandsminimums.

VERTEXEDGEDISTANCE(e_1, a_2)
(1) $\quad \boldsymbol{v}_1 \leftarrow \boldsymbol{b}_1 - \boldsymbol{a}_1$
(2) $\quad v_{12} \leftarrow \boldsymbol{a}_2 - \boldsymbol{a}_1$
(3) $\quad W_1 \leftarrow \boldsymbol{v}_1^T \boldsymbol{v}_{12}$
(4) $\quad V_1 \leftarrow \boldsymbol{v}_1^2$
(5) $\quad$ **if** $(W_1 > 0 \wedge V_1 < 0) \vee (W_1 < 0 \wedge V_1 > 0)$
(6) $\quad\quad$ **return** $\big[\boldsymbol{v}_{12}^2, (a_1, a_2)\big]$
(7) $\quad$ **else if** $(W_1 > V_1 \wedge V_1 > 0) \vee (W_1 < V_1 \wedge V_1 < 0)$
(8) $\quad\quad$ **return** $\big[(\boldsymbol{a}_2 - \boldsymbol{b}_1)^2, (b_1, a_2)\big]$
(9) $\quad$ **else**
(10) $\quad\quad \mu_1^* \leftarrow \frac{W_1}{V_1}$
(11) $\quad\quad$ **return** $\big[\boldsymbol{v}_{12}^2 - \mu_1^* W_1, (\mu_1^*), a_2)\big]$

3.4.3 Punkt-Fläche-Abstand

In Abschnitt 3.4.1 haben wir ein Verfahren zur Bestimmung des Euklidischen Abstands zweier Kanten vorgestellt. Die Beispiele aus Abbildung 3.1 verdeutlichen jedoch, daß ein weiterer Abstandstest zur Ermittlung der Distanz eines disjunkten Flächenpaares (f_1, f_2) erforderlich ist. Aus diesem Grund betrachten wir im folgenden das Problem der Punkt-Fläche-Abstandsberechnung. Wir wollen zunächst in Analogie zur Vorgehensweise im Kante-Kante-Fall die relaxierte Fragestellung untersuchen. Hierbei besteht die Aufgabe in der Ermittlung des Euklidischen Abstands zwischen dem Punkt a und der unterstützenden Ebene von f. Diese bezeichnen wir im folgenden mit $\Sigma(f)$ und schreiben $\Sigma_{\boldsymbol{n}, n_0}$, um explizit auf ihre Darstellung in Hessesche-Normalform hinzuweisen. Es gilt also $\Sigma(f) = \big\{x \in \mathbb{R}^3 \,\big|\, \boldsymbol{n}^T \boldsymbol{x} = n_0\big\}$. Das Vorzeichen des Normalenvektors von f ist dabei so gewählt, daß die in zyklischer Reihenfolge bezeichneten Endpunkte

von f, $b_0, b_1, \ldots, b_{k-1}, b_k \equiv b_0$, im Gegenuhrzeigersinn aufgezählt sind, sofern man in Richtung von $-\mathbf{n}$ auf f blickt. Der skalare Wert n_0 gibt den Abstand der Ebene vom Ursprung an.
Wir erhalten somit die Distanz des Punktes a zu der Ebene $\Sigma(f)$ als

$$\delta\big(a, \Sigma(f)\big) = |\,\mathbf{n}^T\mathbf{a} - n_0\,|\,.$$

Der naheste Punkt von $\Sigma(f)$ zu a entspricht der Projektion $\pi_{\Sigma(f)}$ von a auf die Ebene:

$$\pi_{\Sigma(f)} = \mathbf{a} - (\mathbf{n}^T\mathbf{a} - n_0)\mathbf{n}\,.$$

Als nahestes Punktepaar erhalten wir somit

$$\chi\big(a, \Sigma(f)\big) := (a, \pi_{\Sigma(f)})\,.$$

Unser Ziel ist es jedoch, den Euklidischen Abstand von a zu einer Fläche $f \subset \Sigma(f)$ zu bestimmen. Der Abstand zwischen f und $\Sigma(f)$ stellt lediglich eine untere Schranke für diesen Wert dar.

$$\delta(a, f) \geq \delta\big(a, \Sigma(f)\big)\,.$$

Das naheste Punktepaar zwischen a und f fällt mit dem zwischen a und $\Sigma(f)$ zusammen, falls die Projektion von a auf die Ebene im Innern von f liegt. Befindet sich der Punkt $\pi_{\Sigma(f)}(a)$ dagegen außerhalb des Polygons f, so wird der minimale Abstand zwischen a und einer Kante von f angenommen.

$$\delta(a, f) = \begin{cases} \delta\big(a, \Sigma(f)\big) & \text{falls } \pi_{\Sigma(f)}(a) \in f\,; \\ \min\limits_{e_f \in \mathcal{E}(f)} \delta(a, e_f) & \text{sonst}\,. \end{cases} \tag{3.10}$$

Der naheste Punkt von f zu a kann somit folgendermaßen festgelegt werden:

$$\chi(a, f) := \begin{cases} \chi\big(a, \Sigma(f)\big) & \text{falls } \pi_{\Sigma(f)}(a) \in f\,; \\ \chi(a, e^*) & \text{sonst}\,, \end{cases}$$

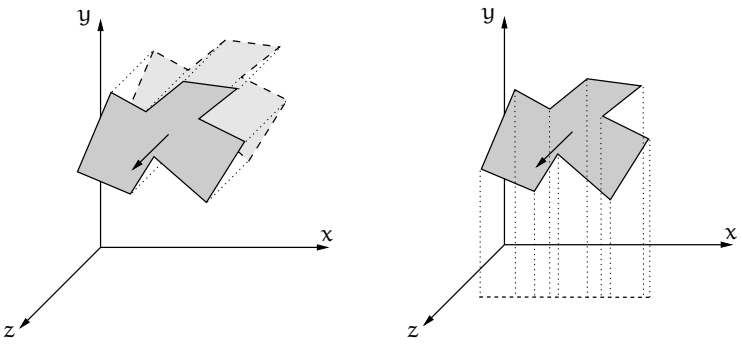
wobei $e^* \in \big\{e \in \mathcal{E}(f) \,\big|\, \delta(a, e) = \min\limits_{e_f \in \mathcal{E}(f)} \delta(a, e_f)\big\}$ beliebig gewählt werden kann.

Das wichtigste Prädikat in diesem Zusammenhang gibt also eine Antwort auf die Frage, ob ein Punkt $p := \pi_{\Sigma(f)}(a)$ innerhalb des Polygons f liegt. Mit dieser Fragestellung werden wir uns im folgenden beschäftigen.

Überführung der Eingabedaten in den $\mathbb{R}^2$

Da die Fragestellung offensichtlich zweidimensionaler Natur ist, wollen wir zunächst das Polygon f und den Anfragepunkt p in den $\mathbb{R}^2$ transformieren, in dem der Punkt-im-Polygon-Test effizient durchgeführt werden kann. Dies erreichen wir, indem wir sowohl den Punkt, als auch das Polygon auf eine der Koordinatenebenen projizieren.

Abbildung 3.12 *Extremfall*: Projektion eines Polygons, das parallel zu einer der Koordinatenebenen ist.



Projektion auf die x-y-Ebene. Projektion auf die x-z-Ebene.

Welche Koordinate dazu ignoriert wird, hängt selbstverständlich von den Eingabedaten ab. Projiziert man ein Polygon stets auf die gleiche Ebene, so läuft man Gefahr, numerisch schlecht konditionierte Daten oder schlimmstenfalls ein zu einer Kante degeneriertes Polygon zu erhalten. Abbildung 3.12 zeigt eine optimale und eine denkbar schlechte Wahl der Projektionsebene.

Das Problem läßt sich jedoch einfach vermeiden, indem man die Koordinatenebene wählt, die „möglichst parallel“ zu f ist. Dazu ist es lediglich notwendig, den betragsmäßig größten Koordinatenindex des Normalenvektors zu identifizieren. Diese Vektorkomponente kann im folgenden ignoriert werden, da die anderen beiden Indizes die Projektionsebene definieren. Es ist nun leicht einzusehen, daß die folgende Bedingung stets erfüllt ist:

$$p \in f \iff \pi(p) \in \pi(f)\,,$$

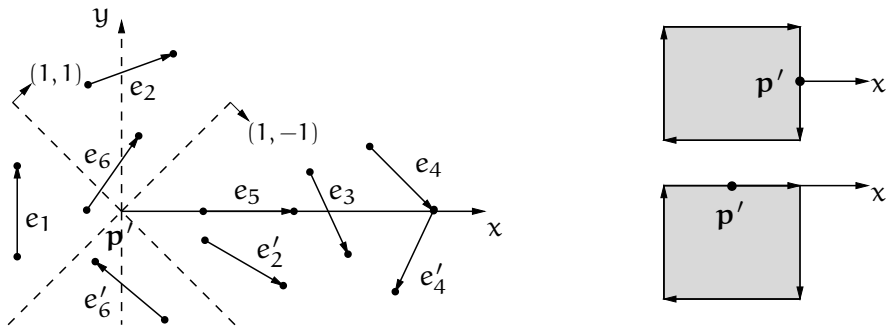
wobei $\pi(p) = p'$ bzw. $\pi(f) = (b'_0, b'_1, \ldots, b'_{k-1})$ die Projektion von p bzw. f auf die oben beschriebene Koordinatenebene bezeichnet. Wir wollen im folgenden annehmen, daß die xy-Ebene als Projektionsebene gewählt wurde. Das so entstandene Polygon $(b'_0, b'_1, \ldots, b'_{k-1})$ wollen wir nun mit f', den Anfragepunkt mit p' bezeichnen.

In der Literatur ist der Punkt-im Polygon-Test in sehr vielen Variationen diskutiert worden. Man vergleiche hierzu [Zac94]. Eine der einfachsten und effizientesten Methoden ist das sogenannte *One-Shot-Verfahren*, welches wir im folgenden vorstellen wollen.

Das One-Shot-Verfahren

Die grundlegende Idee des Verfahrens ist denkbar einfach. Um zu überprüfen, ob ein Punkt p' im Innern des Polygons f' liegt, sendet man einen beliebigen Strahl mit Anfangspunkt p aus und zählt die Zahl der Schnittpunkte mit den Kanten des Polygons. Ist die Zahl der Schnittpunkte ungerade, so liegt der Punkt p' innerhalb des Polygons, andernfalls außerhalb.

Abbildung 3.13 Kantenkonfigurationen, für die der Schnittest mit dem Strahl sehr einfach beantwortet werden kann.



Die Richtung des Strahls ist bei dieser Vorgehensweise beliebig. Auch die Position des Anfragepunktes im Koordinatensystem kann verändert werden, sofern das Polygon die gleiche Translation erfährt. Diese Freiheiten können wir ausnutzen, um die durchzuführenden Schnittests zu vereinfachen. Verschieben wir den Punkt p', mit $\mathbf{p}' = (x_p, y_p)^T$ in den Ursprung des Koordinatensystems und wählen die positive x-Achse als Strahlrichtung, so können wir in fast allen Situationen einen Kante-Strahl-Schnittest durch einfache Koordinatenvergleiche beantworten. Jeder Eckpunkt b'_i des Polygons besitzt nach der Translation die Koordinaten $(x_i - x_p, y_i - y_p)$.

Wir orientieren uns im folgenden an dem Algorithmus von [Zac94], der – einer empirischen Studie des Autors nach – lediglich in 1% aller Fälle den exakten Schnittpunkt zwischen Kante und Strahl berechnen muß. Liegt der Anfragepunkt auf dem Rand des Polygons, so wird diese Situation nicht immer korrekt erkannt und der Punkt in bestimmten Fällen als außerhalb des Polygons gelegen klassifiziert. Diese Einschränkung stellt jedoch kein Problem dar, da eine solche Situation bereits durch den Kante-Kante-Test erfaßt wird.

Die folgende Fallunterscheidung, die sich an der Lage der zu betrachtenden Kante relativ zum Anfragepunkt p orientiert, entscheidet, ob die Kante (b'_i, b'_{i+1}) den Strahl schneidet.

1.Fall: $x_i \leq x_p \wedge x_{i+1} \leq x_p$
Liegen beide Eckpunkte außerhalb der Halbebene, die durch den Strahl als Normalenvektor und p' als Aufpunkt definiert wird, so kann es keinen Schnittpunkt zwischen der Kante und dem Strahl geben. In Abbildung 3.13 erfüllt die Kante e_1 diese Eigenschaft. Befinden sich beide Punkte auf dem Rand der Halbebene so klassifizieren wir den Anfragepunkt ebenfalls als außerhalb des Polygons gelegen. Diese Vorgehensweise stellt eine implizite Berücksichtigung der ε-Toleranz dar, die der Berechnung a priori zugrunde liegt. Daraus folgt jedoch unmittelbar, daß wir die Situation, die im rechten Teil von Abbildung 3.13 dargestellt ist, nicht korrekt erkennen können.

2.Fall: $(y_i > y_p \wedge y_{i+1} > y_p) \vee (y_i < y_p \wedge y_{i+1} < y_p)$
Beide Endpunkte liegen bezüglich der y-Achse entweder oberhalb oder unterhalb des Strahls, so daß ein Schnittpunkt zwischen Kante und Strahl ausgeschlossen werden kann. Die Situation ist in Abbildung 3.13 für e_2 bzw. e_2' dargestellt.

3.Fall: $(x_i > x_p \wedge x_{i+1} > x_p) \wedge \left[(y_i > y_p \wedge y_{i+1} < y_p) \vee (y_i < y_p \wedge y_{i+1} > y_p) \right]$
Liegen beide Endpunkte innerhalb der Halbebene, durch die der Strahl verläuft, wobei sich einer „oberhalb" und der andere unterhalb des Strahls befindet, so muß die zugehörige Kante den Strahl kreuzen. Es liegt also ein Schnittpunkt vor, wie das Beispiel e_3 in Abbildung 3.13 verdeutlicht.

4.Fall: $(x_i > x_p \wedge x_{i+1} > x_p) \wedge (y_i = y_p \vee y_{i+1} = y_p)$
Verläuft der Strahl durch genau einen der beiden Kantenendpunkte, so liegt eine entartete Situation vor. Der Schnittpunkt zwischen den beiden Kanten und dem Strahl darf nur einmal gezählt werden (Kante e_4 und e_4' in Abbildung 3.13). In unserem Algorithmus berücksichtigen wir daher nur solche Schnittpunkte, in denen die Kante „oberhalb" des Strahls verläuft, d.h. wenn gilt: $(y_i = y_p \wedge y_{i+1} > y_p) \vee (y_i > y_p \wedge y_{i+1} = y_p)$. In unserem Beispiel würde also lediglich der Schnittpunkt der Kante e_4 mit dem Strahl gezählt werden.

5.Fall: $(x_i > x_p \vee x_{i+1} > x_p) \wedge (y_i = y_p \wedge y_{i+1} = y_p)$
Im Fall der Kollinearität von Kante und Strahl hängt das Testergebnis von der Frage ab, ob der Anfragepunkt innerhalb oder außerhalb der Kante liegt. Falls die Bedingung $(x_i > x_p \wedge x_{i+1} > x_p)$ gilt, so dürfen wir keinen Schnittpunkt zählen, da p' außerhalb des Polygons liegt, sofern er nicht von einem anderen Kantenzug eingeschlossen wird. Das Beispiel aus Abbildung 3.13 verdeutlicht diese Situation. Überdecken sich Strahl und Kante dagegen teilweise, d.h. gilt $(x_i > x_p \wedge x_{i+1} < x_p) \wedge (x_i < x_p \wedge x_{i+1} > x_p)$, so wollen wir in Analogie zu Fall 1 auch diese Situation als schnittfrei im Rahmen der ε-Toleranz klassifizieren. Auch in diesem entarteten Fall lassen sich Beispiele konstruieren, in der die Vorgehensweise nicht das Resultat eines exakten Tests liefert (vgl. rechtes Beispiel aus Abbildung 3.13).

6.Fall: $\left[(x_i \leq x_p \wedge x_{i+1} \geq x_p) \vee (x_i \geq x_p \wedge x_{i+1} \leq x_p) \right] \wedge$
$\left[(y_i \leq y_p \wedge y_{i+1} \geq y_p) \vee (y_i \geq y_p \wedge y_{i+1} \leq y_p) \right]$
Dieser letzte Fall entspricht den Kantenkonfigurationen e_6 und e_6' in Abbildung 3.13. Offensichtlich kann auch hier ein Schnittpunkt ausgeschlossen werden, wenn eine Kante vollständig in einem der beiden Halbräume liegt, die durch p' und den Vektor $(1,1)^T$ bzw. $(1,-1)^T$ definiert werden. Diese Bedingung ist erfüllt, falls

$$\begin{aligned} & \left[y_i \geq x_i + (y_p - x_p) \wedge y_{i+1} \geq x_{i+1} + (y_p - x_p) \right] \vee \\ & \left[y_i \leq -x_i + (y_p + x_p) \wedge y_{i+1} \leq -x_{i+1} + (y_p + x_p) \right] \\ \iff \quad & \left[y_i - y_p \geq x_i - x_p \wedge y_{i+1} - y_p \geq x_{i+1} - x_p \right] \vee \\ & \left[-(y_i - y_p) \geq x_i - x_p \wedge -(y_{i+1} - y_p) \geq x_{i+1} - x_p \right] \end{aligned}$$

gilt. Der Gebrauch von $\leq$ bzw. $\geq$ stellt auch in diesem Fall eine implizite Berücksichtigung der ε-Toleranz dar.

Andernfalls können wir durch simple Koordinatenvergleiche keine weiteren Schnittests mehr durchführen. Wir sind daher gezwungen den Schnittpunkt der Kante mit der x-Achse zu berechnen und anschließend zu überprüfen, ob die so gefundene Nullstelle x^* auf dem Strahl liegt. Als Schnittpunkt mit der x-Achse ergibt sich:

$$x^* = \frac{(y_{i+1} - y_i)(x_i - x_p) - (x_{i+1} - x_i)(y_i - y_p)}{y_{i+1} - y_i}.$$

Die Kante schneidet somit genau dann den Strahl, wenn $x^* \geq 0$ gilt (x^* nicht explizit berechnen!).

Algorithmus 6 faßt unsere Überlegungen als sogenannten Punkt-in-Polygon-Test zusammen. Dazu transformiert er zunächst die Eingabedaten in den $\mathbb{R}^2$, indem er p und f auf eine geeignete Koordinatenebene projiziert. Anschließend führt er die gerade beschriebenen Tests in einer sorgfältig gewählten Reihenfolge für jede Kante des Polygons aus. und beantwortet auf diese Weise die Frage, ob der Anfragepunkt p im Innern von f liegt.

Algorithmus

Wir haben nun alle Grundlagen gelegt, um ein Verfahren zur Bestimmung des Euklidischen Abstands eines Punktes a von einer Fläche f angeben zu können. Der hier vorgestellte Pseudoalgorithmus 7 berechnet zunächst die Projektion $p = \pi_{\Sigma(f)}$ von a auf die unterstützende Ebene $\Sigma(f)$. Mit Hilfe des Punkt-In-Polygon-Tests wird anschließend überprüft, ob der Anfragepunkt p innerhalb der Fläche f liegt. Entsprechend Gleichung 3.10 bestimmt der Wert dieses Prädikats, wie der Abstand zwischen dem Punkt und der Fläche ermittelt wird. Liegt der Punkt p im Polygon f, so wird der Euklidische Abstand im nahesten Punktepaar (a, p) angenommen. Andernfalls ermitteln wir den minimalen Abstand von a zu allen Kanten von f.

3.4.4 Der Kante-Fläche-Abstand

Wie wir bereits dargestellt haben, erlauben die bisher vorgestellten Elementartests, lediglich die Distanz zweier disjunkter bzw. sich berührender Flächen zu berechnen. Im Überlappungsfall wird der minimale Abstand der beiden Flächen in den Schnittpunkten der beiden Polygone angenommen. Ein solcher Schnittpunkt ist im allgemeinen ein Punkt im Innern der einen Fläche und gleichzeitig ein innerer Punkt einer Kante der anderen Fläche. Beschränken wir uns auf Kante-Kante- und Punkt-Fläche-Tests, so kann diese Durchdringung offensichtlich nicht entdeckt werden. Die Abstandsberechnung liefert entgegen unserer Vereinbarung einen positiven Abstandswert (vgl. Abbildung 3.14). Aus diesem Grund benötigen wir einen weiteren elementaren Test, der überprüft, ob eine Kante der einen Fläche die andere Fläche durchdringt.

Algorithmus 6 Ein Algorithmus, der bestimmt, ob ein Punkt in einem Polygon liegt.

Eingabe: Eine Fläche $f = (b_0, b_1, \ldots, b_{k-1})$ und ein Punkt p im $\mathbb{R}^3$.

Ausgabe: $\begin{cases} \textbf{true} & \text{falls } p \in f\,; \\ \textbf{false} & \text{sonst}\,. \end{cases}$

POINTINPOLYGON(p, f)
(1) $p' \leftarrow \pi(p)$ (* $p' = (x_p, y_p)^T$ *)
(2) $f' \leftarrow \pi(f)$ (* $f' = (b'_0, b'_1, \ldots, b'_{k-1})$, mit $b'_i = (x_i, y_i)^T$ *)
(3) $s \leftarrow 0$
(4) **for** $i \leftarrow 0$ **to** $k-1$
(5) **if** $x_i \leq x_p \wedge x_{i+1} \leq x_p$
(6) **break**
(7) **if** $y_i > y_p \wedge y_{i+1} > y_p$
(8) **break**
(9) **if** $y_i \leq y_p \wedge y_{i+1} \leq y_p$
(10) **break**
(11) **if** $x_i > x_p \wedge x_{i+1} > x_p$
(12) $s \leftarrow s+1$
(13) **if** $y_i - y_p \geq x_i - x_p \wedge y_{i+1} - y_p \geq x_{i+1} - x_p$
(14) **break**
(15) **if** $-(y_i - y_p) \geq x_i - x_p \wedge -(y_{i+1} - y_p) \geq x_{i+1} - x_p$
(16) **break**
(17) $x^* \leftarrow \frac{(y_{i+1}-y_i)(x_i-x_p)-(x_{i+1}-x_i)(y_i-y_p)}{y_{i+1}-y_i}$
(18) **if** $x^* \geq 0$
(19) $s \leftarrow s+1$
(20) **if** $s \equiv 1 \mod 2$ **then return true**
(21) **else return false**

Der Kante-Fläche-Überlappungstest

Sei $e = \{a + \mu v \mid 0 \leq \mu \leq 1\}$ eine Kante mit zugehöriger Geraden $g = \{a + \lambda v \mid \lambda \in \mathbb{R}\}$ und f eine Fläche mit unterstützender Ebene $\Sigma(f) = \{x \in \mathbb{R}^3 \mid n^T x = n_0\}$. Dann schneidet die Kante e die Fläche f genau dann, wenn die Gerade g die Ebene $\Sigma(f)$ durchdringt und der Schnittpunkt innerhalb von e und f liegt. Wir können somit die beiden folgenden Fälle unterscheiden.

1. Fall: $n \perp v \iff n^T v = 0$
   Falls der Normalenvektor der Ebene senkrecht zu dem Richtungsvektor der Geraden steht, so sind g und $\Sigma(f)$ bzw. e und f parallel. Es existiert daher nur dann ein Schnittpunkt, wenn e in der Ebene $\Sigma(f)$ liegt und dabei eine Kante der Fläche f schneidet oder sich vollständig innerhalb von f befindet. Diese Bedingungen können mit den uns zur Verfügung stehenden Mitteln sehr einfach überprüft werden.

Algorithmus 7 Ein Algorithmus zur Bestimmung des Punkt-Fläche-Abstands.

Eingabe: Ein Punkt a und eine Fläche f mit unterstützender Ebene $\Sigma_{\mathbf{n},n_0}$ in Hessesche Normalform.
Ausgabe: Der quadratische Euklidische Abstand zwischen a und f sowie zwei naheste Punkte als Zeugen des Abstandsminimums.

```
VERTEXFACEDISTANCE(a, f)
(1)   d_{a,Σ(f)} ← n^T a − n_0
(2)   p ← a − d_{a,Σ(f)} n
(3)   if POINTINPOLYGON(p, f)
(4)     return [d²_{a,Σ(f)}, (a, p)]
(5)   else
(6)     d_{a,e*} ← ∞
(7)     foreach e_f ∈ E(f)
(8)       [d_{a,e_f}, (a, p)] ← VERTEXEDGEDISTANCE(e_f, a)
(9)       if d_{a,e_f} < d_{a,e*}
(10)        d_{a,e*} ← d_{a,e_f}
(11)        p* ← p
(12)    return [d_{a,e*}, (a, p*)]
```

Offensichtlich liegt e genau dann in der unterstützenden Ebene von f, falls gilt:

$$\delta\big(a, \Sigma(f)\big) = 0 \quad \Longleftrightarrow \quad \mathbf{n}^T\mathbf{a} - n_0 = 0\,.$$

Ist dieses Prädikat erfüllt, so schneidet die Kante das Polygon f in einer Kante e_f oder in einem beliebigen Punkt, der gleichzeitig ein Punkt von e ist, d.h.

$$\begin{aligned} e \cap f \neq \emptyset \quad &\Longleftrightarrow \quad \big[\exists e_f \in \mathcal{E}(f) : \delta(e, e_f) = 0\big] \ \vee \ e \subset f \\ &\Longleftrightarrow \quad \big[\exists e_f \in \mathcal{E}(f) : \delta(e, e_f) = 0\big] \ \vee \ a \in f\,. \end{aligned}$$

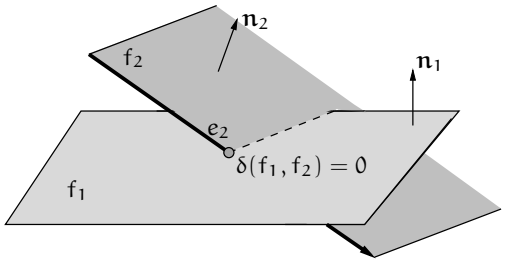
2. Fall: $\mathbf{n} \not\perp \boldsymbol{\nu} \quad \Longleftrightarrow \quad \mathbf{n}^T\boldsymbol{\nu} \neq 0$
Falls Gerade und Ebene nicht parallel sind, existiert ein Schnittpunkt von g und $\Sigma(f)$, der folgendermaßen parametrisiert werden kann:

$$\lambda^* = \frac{n_0 - \mathbf{n}^T\mathbf{a}}{\mathbf{n}^T\boldsymbol{\nu}}\,. \tag{3.11}$$

Der Punkt x^* mit $\mathbf{x}^* = a + \lambda^*\boldsymbol{\nu}$ ist genau dann ein Schnittpunkt von e und f, wenn $\lambda^* \in [0,1]$ und $x^* \in f$ gilt. In vielen Fällen ist es möglich auf die Fließkommadivision in Gleichung 3.11 zu verzichten, da das Prädikat $\lambda^* \in [0,1]$ durch Vergleich von Zähler und Nenner effizient ausgewertet werden kann. Sei $s := n_0 - \mathbf{n}^T\mathbf{a}$ und $t := \mathbf{n}^T\boldsymbol{\nu}$, dann gilt:

(i) $\lambda^* \in [0,1] \quad \Longleftrightarrow \quad (0 \leq s \leq t) \ \vee (t \leq s \leq 0)$

Abbildung 3.14 Im Überlappungsfall wird der Wert $\delta(f_1, f_2)$ weder an einem Kante-Kante- noch an einem Punkt-Fläche-Paar angenommen. Es ist daher ein Kante-Fläche-Schnittest erforderlich, um die Flächenkonfiguration korrekt behandeln zu können.



(ii) $\lambda^* \notin [0, 1]$

Ein effizientes Verfahren, das entscheidet, ob $x^* \in f$ liegt haben wir bereits in Abschnitt 3.4.3 vorgestellt. Anhand dieser Überlegungen ergibt sich ein einfacher Kante-Fläche-Überlappungstest, der in Algorithmus 8 dargestellt ist.

Der Kante-Fläche-Abstandstest

Führen wir die unabhängig voneinander entwickelten elementaren Abstands- und Überlappungstests zu einem Kante-Fläche-Abstandsberechnungsverfahren zusammen, so ergibt sich weiteres Optimierungspotential.

Im Fall des Punkt-Fläche-Tests haben wir festgestellt, daß der minimale Abstand zwischen dem Punkt und einer Kante der Fläche angenommen wird, falls die Projektion des Punktes auf die unterstützende Ebene der Fläche außerhalb letzterer liegt. Falls ein solcher Punkt-Kante-Test die Distanz der beiden Flächen definiert, so erhalten wir den gesuchten Abstand durch einen der durchgeführten Kante-Kante-Tests. Schließlich ist der betrachtete Punkt Endpunkt einer Kante der Fläche. In der Gesamtbetrachtung können wir also auf die Punkt-Kante-Abstandsberechnung in Algorithmus 5 (Zeile 6 bis 12) verzichten.

Ein ähnliche Argumentation läßt sich auch für den Kante-Fläche-Überlappungstest (Algorithmus 8) führen. Liegt die Kante innerhalb der unterstützenden Ebene der Fläche, so kann auf eine explizite Behandlung dieser Situation (Zeile 6 bis 11) verzichtet werden, da im Kante-Fläche-Überlappungsfall ein Kante-Kante- bzw. Punkt-Fläche-Abstandstest den korrekten Distanzwert von 0 liefert.

Nach Beseitigung dieser redundanten Berechnungen in den beiden Elementartests, ergibt sich ein einfaches Kante-Fläche-Abstandsberechnungsverfahren, das durch Algorithmus 9 beschrieben wird.

Vergleicht man die Berechnungen, die im Rahmen der Punkt-Kante-Abstandsberechnung durchgeführt werden, mit den erforderlichen Berechnungen im Kante-

Algorithmus 8 Ein statischer Überlappungstest zwischen einer Kante und einer Fläche.

Eingabe: Eine Kante $e = (a, b)$ und eine Fläche f mit unterstützender Ebene $\Sigma_{\mathbf{n},n_0}$ in Hessesche Normalform.

Ausgabe: $\begin{cases} [\textbf{true}\,, p] & \text{falls } e \cap f \neq \emptyset \wedge p \in e \cap f\,; \\ [\textbf{false}\,, \textbf{nil}\,] & \text{sonst}\ . \end{cases}$

EDGEFACEINTERSECTION(e, f)

(1) $\quad \boldsymbol{\nu} \leftarrow \mathbf{b} - \mathbf{a}$
(2) $\quad t \leftarrow \mathbf{n}^T\boldsymbol{\nu}$
(3) $\quad$ **if** $t = 0 \qquad$ (* $e \parallel f$ *)
(4) $\quad\quad$ **if** $\mathbf{n}^T\mathbf{a} - n_0 \neq 0 \qquad$ (* $e \not\subset f$ *)
(5) $\quad\quad\quad$ **return** [**false** , **nil**]
(6) $\quad\quad$ **foreach** $e_f \in \mathcal{E}(f)$
(7) $\quad\quad\quad \left[d, (p_1, p_2)\right] \leftarrow$ EDGEEDGEDISTANCE(e, e_f)
(8) $\quad\quad\quad$ **if** $d = 0$
(9) $\quad\quad\quad\quad$ **return** [**true** , p_1]
(10) $\quad\quad$ **if** POINTINPOLYGON$(\mathbf{a}, f)$ **then return** [**true** , a]
(11) $\quad\quad\quad\quad\quad\quad\quad\quad$ **else return** [**false** , **nil**]
(12) $\quad s \leftarrow n_0 - \mathbf{n}^T a$
(13) $\quad$ **if** $t < 0 \wedge (s > 0 \vee s < t) \vee t > 0 \wedge (s < 0 \vee s < t)$
(14) $\quad\quad$ **return** $\left[\textbf{false}\,, \textbf{nil}\,\right] \qquad$ (* $\lambda^* \notin [0, 1]$ *)
(15) $\quad \boldsymbol{x}^* \leftarrow \mathbf{a} + \frac{s}{t}\boldsymbol{\nu}$
(16) $\quad$ **if** POINTINPOLYGON$(\boldsymbol{x}^*, f)$ **then return** [**true** , x^*]
(17) $\quad\quad\quad\quad\quad\quad\quad\quad$ **else return** [**false** , **nil**]

Fläche-Überlappungstest, so ergeben sich weitere Redundanzen, die durch die Kombination der beiden Tests beseitigt werden können. Zunächst stellen wir fest, daß der in Zeile 1 des Punkt-Fläche-Algorithmus ermittelte Abstand des Anfragepunktes a von der Ebene $\Sigma(f)$ betragsmäßig mit dem Zähler s des Kante-Fläche-Schnittpunktparameters $\lambda^* = \frac{s}{t}$ übereinstimmt. Es genügt daher s einmal zu berechnen und anschließend bei Bedarf wiederzuverwenden. Auch der Nenner t des Schnittpunktparameters kann zur Beschleunigung des Punkt-Fläche-Tests eingesetzt werden. Sind f_1 und f_2 die beiden zugrundeliegenden Flächen, dann wird die übergeordnete Fläche-Fläche-Abstandsroutine über alle Kanten e_1 von f_1 sowie e_2 von f_2 iterieren und mit Hilfe der Kante-Fläche-Abstandstests die Distanz der beiden Flächen bestimmen. Dabei wird der ausschließlich für den Anfangspunkt der Kante der Abstand zu Fläche bestimmt, so daß über alle Kante-Fläche-Tests gesehen jeder Eckpunkt der beiden Polygone genau einmal im Rahmen der Punkt-Fläche-Abstandsberechnung berücksichtigt wird. Mit ein wenig Unterstützung der übergeordneten Fläche-Fläche-Abstandsroutine können wir die Zahl der insgesamt durchgeführten Punkt-Fläche-Tests halbieren. Dies sehen wir leicht ein, wenn wir uns überlegen, daß der naheste Eckpunkt der Kante e zu der Fläche

Algorithmus 9 Ein Algorithmus zur Bestimmung des Kante-Fläche-Abstands.

Eingabe: Eine Kante $e = (a, b)$ und eine Fläche f mit unterstützender Ebene $\Sigma_{\mathbf{n},n_0}$ in Hessesche Normalform.
Ausgabe: Der quadratische Euklidische Abstand zwischen e und f sowie zwei naheste Punkte als Zeugen des Abstandsminimums.

```
EDGEFACEDISTANCE(e, f)
(1)   [b, p] ← EDGEFACEINTERSECTION(e, f)
(2)   if b = true
(3)     return [0, (p, p)]
(4)   d*_ef ← ∞
(5)   [d, (p_1, p_2)] ← VERTEXFACEDISTANCE(a, f)
(6)   if d < d*_ef
(7)     d*_ef ← d
(8)     p*_i ← p_i,   i = 1, 2
(9)   foreach e_f ∈ E(f)
(10)    [d, (p_1, p_2)] ← EDGEEDGEDISTANCE(e, e_f)
(11)    if d < d*_ef
(12)      d*_ef ← d
(13)      p*_i ← p_i,   i = 1, 2
(14)    return [d*_ef, (p*_1, p*_2)]
```

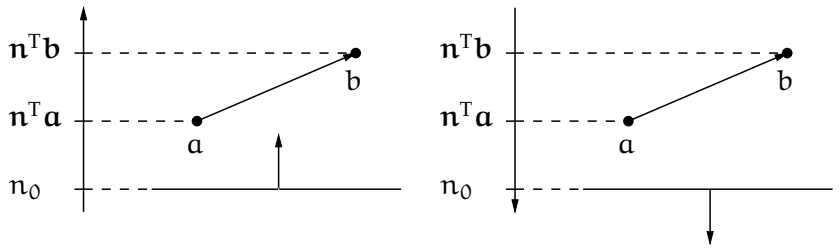
f anhand der Werte s und t bestimmt werden kann. Dabei können wir den Durchdringungsfall ausklammern, da s und t ja gerade den Schnittpunkt definieren und unsere Behauptung in diesem Fall trivialerweise erfüllt ist.

Das Prädikat $s = n_0 - \mathbf{n}^T\mathbf{a} \gtrless 0$ entscheidet, ob sich der Punkt a „unterhalb“ bzw. „oberhalb“ der unterstützenden Ebene $\Sigma(f) = \left\{\mathbf{x} \in \mathbb{R}^3 \,\middle|\, \mathbf{n}^T\mathbf{x} = n_0\right\}$ befindet. Abhängig von der Lage des Punktes bezüglich dieser Ebene können wir mit Hilfe des Vergleichs $t = \mathbf{n}^T\boldsymbol{\nu} = \mathbf{n}^T\mathbf{b} - \mathbf{n}^T\mathbf{a} \gtrless 0$ bestimmen, welcher Eckpunkt von e am nahesten zu $\Sigma(f)$ liegt. Unter der Voraussetzung $e = (a, b) \cap f = \emptyset$ gilt dabei:

$$
\begin{aligned}
\delta\big(\{a, b\}, \Sigma(f)\big) &= \begin{cases}
\delta(a, f) & \text{falls } (\mathbf{n}^T\mathbf{a} > n_0 \wedge \mathbf{n}^T\mathbf{b} > \mathbf{n}^T\mathbf{a}) \vee \\
 & \qquad (\mathbf{n}^T\mathbf{a} < n_0 \wedge \mathbf{n}^T\mathbf{b} < \mathbf{n}^T\mathbf{a})\,; \\
\delta(b, f) & \text{falls } (\mathbf{n}^T\mathbf{a} > n_0 \wedge \mathbf{n}^T\mathbf{b} < \mathbf{n}^T\mathbf{a}) \vee \\
 & \qquad (\mathbf{n}^T\mathbf{a} < n_0 \wedge \mathbf{n}^T\mathbf{b} > \mathbf{n}^T\mathbf{a})\,; \\
\delta(a, f) = \delta(b, f) & \text{sonst}\,.
\end{cases} \\
&= \begin{cases}
\delta(a, f) & \text{falls } s < 0 \wedge t > 0\,; \\
\delta(b, f) & \text{falls } s > 0 \wedge t < 0\,; \\
\delta(a, f) = \delta(b, f) & \text{sonst}\,.
\end{cases}
\end{aligned}
$$

Abbildung 3.15 veranschaulicht die Aussage. Für die Punkt-Fläche-Abstandsberech-

Abbildung 3.15 Die beiden Fälle, in denen a am nahesten zu f liegt.



nung wählen wir den Eckpunkt mit minimalem Abstand zur unterstützenden Ebene $\Sigma(f)$. O.B.d.A. sei dies der Punkt a. Der Unterschied zu unseren bisherigen Überlegungen liegt nun darin, daß die Projektion von a auf die Fläche f nicht zwangsläufig innerhalb der Fläche liegt und es somit möglich ist, daß b einen geringeren Abstand zu f aufweist als a. In diesem Fall existiert jedoch ein Punkt auf der Kante e, der, wie man sich leicht überlegen kann, näher zu f liegt als a und b und somit im Rahmen eines Kante-Kante-Tests berücksichtigt wird. Da wir nun bei der Kante-Fläche-Abstandsberechnung beide Eckpunkte der Kante betrachten, genügt es, die Punkt-Fläche-Tests für jede zweite Kante durchzuführen. Der Kante-Fläche-Test kann jedoch nicht wissen, welche Kante er wie behandeln soll, so daß wir als zusätzlichen Eingabeparameter eine Boolsche Variable b benötigen, deren Wert anzeigt, ob die Punkt-Fläche-Abstandsberechnung für die Kante aufgerufen werden soll oder nicht. Der optimierte Kante-Fläche-Test ist in Algorithmus 10 beschrieben.

3.4.5 Der Fläche-Fläche-Abstand

In Abschnitt 3.1 haben wir gesehen, daß das Problem der Bestimmung des Euklidischen Abstands zweier Flächen auf die Abstandsberechnung zwischen Kanten- und Punkt-Fläche-Paaren sowie einen Kante-Fläche-Überlappungstest reduziert werden kann. Diese drei elementaren Problemstellungen haben wir in diesem Abschnitt betrachtet und effiziente Verfahren zu deren Lösung vorgestellt. Im vorangegangenen Abschnitt haben wir die Elementartests um redundante Berechnungen bereinigt. Das Ergebnis haben wir in Form eines Kante-Fläche-Abstandsberechnungsverfahrens formuliert. Die Aufgabe der Fläche-Fläche-Distanzroutine besteht nun lediglich darin, die Paare der zu testenden Kante-Fläche-Paare aufzuzählen und jede zweite Kante durch das Setzen einer Boolschen Variablen b zu markieren. Im Rahmen der Kante-Fläche-Tests, für die b den Wert **true** annimmt, wird auf den Aufruf der Punkt-Fläche-Abstandsberechnung verzichtet (vgl. Algorithmus 10). Somit ergibt sich das durch Algorithmus 11 beschriebene Verfahren, das wir als elementaren Abstandstest unserem Branch-and-Bound-Verfahren zugrundelegen wollen.

Algorithmus 10 Ein Algorithmus zur Bestimmung des Kante-Fläche-Abstands.

Eingabe: Eine Kante $e = (a, b)$, eine Fläche f mit unterstützender Ebene $\Sigma_{\mathbf{n},n_0}$ in Hessesche Normalform sowie eine Boolsche Variable b, mit

$$b = \begin{cases} \textbf{true} & \text{falls Punkt-Fläche-Test durchzuführen ist ;} \\ \textbf{false} & \text{sonst .} \end{cases}$$

Ausgabe: Der quadratische Euklidische Abstand zwischen e und f sowie zwei naheste Punkte als Zeugen des Abstandsminimums.

EDGEFACEDISTANCE(e, f, b)
(1) $\quad \boldsymbol{\nu} \leftarrow \mathbf{b} - \mathbf{a}$
(2) $\quad s \leftarrow n_0 - \mathbf{n}^T a$
(3) $\quad t \leftarrow \mathbf{n}^T \boldsymbol{\nu}$
(4) $\quad$ **if** $t \neq 0 \qquad$ (* $e \nparallel f$ *)
(5) $\quad\quad$ **if** $(0 \leq s \wedge s \leq t) \vee (t \leq s \wedge s \leq 0) \qquad$ (* $\lambda^* \in [0, 1]$ *)
(6) $\quad\quad\quad \mathbf{x}^* \leftarrow \mathbf{a} + \frac{s}{t}\boldsymbol{\nu}$
(7) $\quad\quad\quad$ **if** POINTINPOLYGON$(\mathbf{x}^*, f)$
(8) $\quad\quad\quad\quad$ **return** $\left[0, (x^*, x^*)\right]$
(9) $\quad d^*_{ef} \leftarrow \infty$
(10) $\quad$ **if** $b =$ **true** $\qquad$ (* Punkt-Fläche-Test durchführen *)
(11) $\quad\quad$ **if** $s < 0 \wedge t > 0 \qquad$ (* a nahester Punkt *)
(12) $\quad\quad\quad \mathbf{p}_1 \leftarrow \mathbf{a}$
(13) $\quad\quad$ **else if** $s > 0 \wedge t < 0 \qquad$ (* b nahester Punkt *)
(14) $\quad\quad\quad \mathbf{p}_1 \leftarrow \mathbf{b}$
(15) $\quad\quad \mathbf{p}_2 \leftarrow \mathbf{p}_1 + s\mathbf{n}$
(16) $\quad\quad$ **if** POINTINPOLYGON$(\mathbf{p}_2, f)$
(17) $\quad\quad\quad s \leftarrow |s|$
(18) $\quad\quad\quad$ **if** $s < d^*_{ef}$
(19) $\quad\quad\quad\quad d^*_{ef} \leftarrow s$
(20) $\quad\quad\quad\quad p^*_i \leftarrow p_i, \quad i = 1, 2$
(21) $\quad\quad$ **foreach** $e_f \in \mathcal{E}(f)$
(22) $\quad\quad\quad \left[d, (p_1, p_2)\right] \leftarrow$ EDGEEDGEDISTANCE(e, e_f)
(23) $\quad\quad\quad$ **if** $d < d^*_{ef}$
(24) $\quad\quad\quad\quad d^*_{ef} \leftarrow d$
(25) $\quad\quad\quad\quad p^*_i \leftarrow p_i, \quad i = 1, 2$
(26) $\quad\quad\quad$ **return** $\left[d^*_{ef}, (p^*_1, p^*_2)\right]$

3.5 Hüllkörperhierarchien

3.5.1 Innere und äußere Körperapproximation

Das von uns vorgestellte Verfahren verwendet eine approximierte Darstellung der Körper bzw. bestimmter Teile von diesen, um die Abstandsberechnung zu vereinfachen

Algorithmus 11 Ein Algorithmus zur Bestimmung des Fläche-Fläche-Abstands.

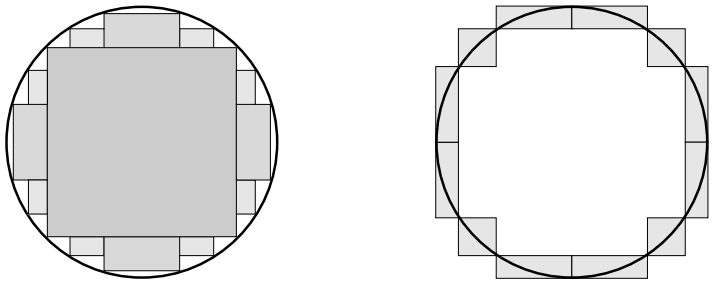
Eingabe: Zwei Flächen f_1 und f_2 der beiden Körper K_1 und K_2.
Ausgabe: Der quadratische Euklidische Abstand zwischen f_1 und f_2 sowie zwei naheste Punkte als Zeugen des Abstandsminimums.

FACEFACEDISTANCE(f_1, f_2)
(1) $d^*_{f_1 f_2} \leftarrow \infty$
(2) $j \leftarrow 0$
(3) $b \leftarrow$ **true**
(4) **foreach** $e_1 \in \mathcal{E}(f_1)$
(5) $\quad \left[d, (p_1, p_2)\right] \leftarrow$ EDGEFACEDISTANCE(e_1, f_2, b)
(6) $\quad$ **if** $d = 0$
(7) $\quad\quad$ **return** $\left[d, (p_1, p_2)\right]$
(8) $\quad$ **if** $d < d^*_{f_1 f_2}$
(9) $\quad\quad d^*_{f_1 f_2} \leftarrow d, \quad p^*_i \leftarrow p_i,\ i = 1, 2$
(10) $\quad j \leftarrow j + 1$
(11) $\quad$ **if** $j \equiv 0 \bmod 2$ **then** $b \leftarrow$ **false**
(12) $\quad\quad\quad\quad$ **else** $b \leftarrow$ **true**
(13) **foreach** $e_2 \in \mathcal{E}(f_2)$
(14) $\quad \left[d, (p_1, p_2)\right] \leftarrow$ EDGEFACEDISTANCE(e_2, f_1, b)
(15) $\quad$ **if** $d = 0$
(16) $\quad\quad$ **return** $\left[d, (p_1, p_2)\right]$
(17) $\quad$ **if** $d < d^*_{f_1 f_2}$
(18) $\quad\quad d^*_{f_1 f_2} \leftarrow d, \quad p^*_i \leftarrow p_i,\ i = 1, 2$
(19) $\quad j \leftarrow j + 1$
(20) $\quad$ **if** $j \equiv 0 \bmod 2$ **then** $b \leftarrow$ **false**
(21) $\quad\quad\quad\quad$ **else** $b \leftarrow$ **true**
(22) **return** $\left[d^*_{f_1 f_2}, (p_1, p_2)\right]$

und letztendlich zu beschleunigen. Da eine Approximation durch Primitive im allgemeinen keine exakte Repräsentation der durch die Flächenmenge begrenzten Teilkörper darstellt, handelt es sich bei dem so gefundenen Abstand lediglich um eine Näherungslösung für den tatsächlichen Distanzwert.

Je nachdem, ob wir die (Teil)Körper von „innen“ bzw. von „außen“ (vgl. Abbildung 3.16) approximieren, liefert der Abstand zweier Approximationen eine obere bzw. untere Schranke für die Distanz der angenäherten Volumina. Das von uns in Abschnitt 3.3.4 vorgestellte Branch-and-Bound-Verfahren benötigt jedoch eine untere Abschätzung des tatsächlichen Abstands, um Teilkörperpaare als Kandidaten für das Abstandsminimum ausschließen zu können. Wir möchten an dieser Stelle jedoch darauf hinweisen, daß die Betrachtung der *dualen* Problemstellung, nämlich die Minimierung der oberen Abstandsschranke über eine hierarchische Approximation des Körperinneren, eine berechtigte alternative Vorgehensweise darstellt.

Abbildung 3.16 „Innere“ und „äußere“ Approximation einer Kugel durch achsenorientierte Boxen.



Dennoch haben wir uns für den „klassischen“ Weg entschieden, indem wir die Körper von „außen“, d.h. durch Verpackung ihres Randes annähern. Der Grund für diese Entscheidung liegt in der Einsicht begründet, daß das Abstandsberechnungsproblem nicht isoliert betrachtet werden darf. In einer Dynamiksimulation, in der dem Benutzer ein Eingreifen in das Geschehen gestattet ist, kann die auf Abstandsberechnung beruhende Kollisionserkennung ihre Vorteile nicht ausspielen. Hier sind andere Verfahren gefragt, wie sie beispielsweise in [Eck98] vorgestellt werden. Im Zusammenhang mit diesen Fragestellungen ergibt selbstverständlich nur eine *konservative Approximation* des Körpers Sinn, weil sie die Formulierung eines *hinreichenden Kriteriums für Kollisionsfreiheit* der benutzerinduzierten Bewegung erlaubt. Eine Approximation des Körperinneren führt dagegen zu einem *hinreichenden Kriterium für das Auftreten einer Kollision*. Da die Speicherung einer Approximationshierarchie speicherplatzaufwendig ist, sollten daher Kollisionserkennung und Abstandsberechnung auf der gleichen Datenstruktur operieren.

Eine solche konservative Approximation, bei der also das Volumen des Originalkörpers vollständig in der angenäherten Darstellung enthalten ist, bezeichnen wir im allgemeinen als *Hüllkörper*.

Definition 3.2. Sei H ein Körper und P eine endliche Punktemenge, dann heißt H Hüllkörper von P, falls $H \supseteq P$ gilt. □

Tatsächlich interessieren wir uns weder beim statischen Kollisionserkennungsnoch beim Abstandsberechnungsproblem für eine Verpackung des gesamten Körpers. Bei beiden Problemen ist es ausreichend die jeweilige Fragestellung für den Rand der Körper beantworten zu können (vgl. Beobachtung 1). Dennoch sollten wir die Ausgangsidee der Volumenumhüllung als Anschauung in Erinnerung behalten.

3.5.2 Primitive als Hüllkörpertypen

Zwar können wir jeden von uns modellierten Körper durch eine Vereinigung konvexer Polyeder darstellen, doch eine entsprechende Zerlegung des Körpers zu finden ist

selbst ein anspruchsvolles Ziel und führt häufig zu einer großen Zahl von Polyedern. Wir wollen daher auf sogenannte *Primitive* als Hüllkörper zurückgreifen. Ein Primitiv ist ein konvexes Volumen, das sich durch eine feste Anzahl skalarer Parameter beschreiben läßt [Hen91].
Als wichtigste Vertreter lassen sich nennen:

- Kugeln [Hub95, PG95, Qui94]
- beliebig orientierte Boxen (OBB) [GLM96]
- k-DOPS oder FDH_k [HKM+96, KZ97, Zac98, Kon98]
  Spezialfall: achsenorientierte Boxen (AABB) oder Iso-Boxen [ZF95, Zac97, Zac98]
- Kugelkappen (Sphere-Shells) [KPLM98]

3.5.3 Messung der Approximationsgüte von Hüllkörpern

Offensichtlich haben die vorgestellten Hüllkörper sehr unterschiedliche Approximationseigenschaften, so daß wir diesen Punkt etwas genauer betrachten sollten. Die von uns verfolgte Approximationsstrategie wirft die folgenden Problemstellungen auf, welche die Notwendigkeit der Einführung eines Gütemaßes zur Messung des Approximationsfehler verdeutlichen:

1. Bestimme den *Hüllkörpertyp* der Approximationshierarchie als denjenigen, der hinsichtlich des Gütemaßes die besten Approximationseigenschaften besitzt.
2. Bestimme für eine gegebene Flächenmenge (Teilkörper) und einen vorgegebenen Hüllkörpertyp den optimalen *Hüllkörper* bezüglich des Gütemaßes.

Das grundsätzliche Problem bei der Bewertung von Hüllkörpern ist, daß die Güte der Approximation nicht nur zwischen den einzelnen Typen von Primitiven variiert, sondern auch von der Form des zu approximierenden Körpers und seiner Orientierung abhängt. Es gestaltet sich somit äußerst schwierig, allgemeingültige Aussagen über die *situationsunspezifische Genauigkeit der Approximation* zu machen, wie sie beispielsweise durch Fragestellung 1 impliziert wird.

Für einen gegebenen Körper in *spezifizierter Lage*, lassen sich jedoch geometrische Maße angeben, mit Hilfe derer die Approximationsgüte von Hüllkörpertypen oder konkreten Instanzen verglichen werden können. Somit sind wir zumindest theoretisch in der Lage, die beiden aufgeworfenen Problemstellungen zu lösen, indem wir nach jedem Bewegungsschritt den optimalen Hüllkörpertyp für die aktuelle Orientierung des Objekts ermitteln und anschließend die Hierarchie mit optimalen Hüllkörpern aufbauen. Diese Vorgehensweise ist jedoch mit unserer Aufgabenstellung, eine echtzeitfähige Abstandsberechnung zu entwerfen, unvereinbar. Die Berechnung der Hüllkörper und der Aufbau der Hierarchie sollte offline geschehen, da in der Vorberechnungsphase die Laufzeitanforderungen nicht derart restriktiv wie in der Simulationsphase sind. Dennoch ist es unabdingbar, die bei der Abstandsberechnung betrachteten Hüllkörperpaare entsprechend der aktuellen Objektlage anzupassen (Abschnitt 3.7.2).

Die Dissertation von ECKSTEIN [Eck98] gibt hinsichtlich potentieller Gütemaße einen weitreichenden Überblick. Im Rahmen der Entwicklung einer echtzeitfähigen Kollisionserkennung werden in der Arbeit die Approximationseigenschaften von Kugeln, sowie Iso- und beliebig orientierten Boxen, anhand der folgenden *statisch-geometrischen Gütemaße* untersucht:

1. Volumen,
2. Durchmesser,
3. Oberfläche,
4. gerichteter Hausdorff-Abstand.

Im Kontext der Abstandsberechnung erscheinen die Kriterien *Volumendifferenz* sowie *Hausdorff-Abstand* zwischen umhüllten Körper und Primitiv zur Ermittlung des Approximationsfehlers und somit als Optimierungskriterium bei der Berechnung der Hüllkörper besonders geeignet. Auch wenn wir unserer Intuition folgend von Volumina oder Teilkörpern sprechen, dürfen wir an dieser Stelle nicht vergessen, daß wir uns darauf beschränkt haben, den Rand des Körpers zu verpacken. Ist jedoch eine Flächenmenge $\mathcal{F}$ gegeben, so macht die Definition einer Volumendifferenz keinen Sinn, da nicht garantiert werden kann, daß die Punktmenge den Hüllkörper tatsächlich in Körperäußeres und -inneres partitioniert. Wir müssen daher auf das Gesamtvolumen des Hüllkörpers $V(H)$ als Gütemaß zurückgreifen.

Betrachten wir nun den gerichteten Hausdorff-Abstand von dem Hüllkörper zur eingeschlossenen Flächenmenge:

Definition 3.3 (Gerichteter Hausdorff-Abstand). Seien A und B zwei Mengen, auf denen ein Abstandsmaß d definiert ist, dann heißt

$$d_H(A, B) := \max_{a \in A} \min_{b \in B} d(a, b)$$

gerichteter Hausdorff-Abstand von A zu B. □

Anschaulich bedeutet der gerichtete Hausdorff-Abstand, daß es kein Element $a \in A$ gibt, dessen minimaler Abstand zu B größer als $d_H(A, B)$ ist. Übertragen auf unsere Fragestellung können wir folgern, daß es keinen Punkt des Hüllkörpers H von $\mathcal{F}$ gibt, dessen minimaler Abstand zur Flächenmenge $\mathcal{F}$ größer als $d_H(H, \mathcal{F})$ ist. Beim gerichteten Hausdorff-Abstand handelt es sich somit um eine *worst-case-Abschätzung* des Fehlers, der bei der Abstandsberechnung mittels Hüllkörperapproximation auftreten kann. Doch auch für dieses Gütemaß ergeben sich Probleme in der konkreten Anwendung. Die Berechnung des Hüllkörper mit minimalem Hausdorff-Abstand führt zu nichtkonvexen Optimierungsproblemen, die zudem mit numerischen Methoden schlecht gelöst werden können, weil die Ermittlung des Zielfunktionswertes durch die Approximation des gerichteten Hausdorff-Abstands zeitaufwendig ist. In [Eck98] werden jedoch Verfahren vorgestellt, die für die Hüllkörpertypen Kugel, Iso-Box sowie Quader beliebig

genaue Approximationen des gerichteten Hausdorff-Abstands zur umhüllten Flächenmenge berechnen.

Wir wollen uns im folgenden auf das Primitivvolumen als Optimierungsziel bei der Berechnung der Hüllkörper beschränken und verschiedene Hüllkörpertypen bezüglich dieses Gütemaßes gegenüberstellen.

3.5.4 Die Berechnung volumenminimaler Hüllkörper

Die einschließende Kugel minimalen Volumens

Als Gütemaß für die Approximationseigenschaften eines Hüllkörpers haben wir dessen Volumen gewählt. Das Volumen einer Kugel $S(\mathbf{c}, r)$ mit Mittelpunkt c und Radius r ergibt sich als:

$$V(S) = \frac{4}{3}\pi r^3 .$$

Somit ist die Zielsetzung der Volumenminimierung äquivalent zur *Minimierung des Kugelradius*. Die Restriktion des Optimierungsproblems besteht darin, daß die von uns gesuchte Kugel $S^*(\mathcal{F}) = S(\mathbf{c}^*, r^*)$ eine vorgegebene Flächenmenge $\mathcal{F}$ umschließen soll. Da es sich bei den Flächen aus $\mathcal{F}$ um Polygone handelt, ist leicht einzusehen, daß wir uns darauf beschränken können, die Eckpunkte der konvexen Hülle $CH(\mathcal{V}(\mathcal{F}))$ bzw. die Knoten der gesamten Flächenmenge $\mathcal{V}(\mathcal{F})$ einzuhüllen. Im Rahmen unseres Optimierungsproblems suchen wir also einen Kugelmittelpunkt c^*, so daß der größte Abstand r^* eines Punktes aus $\mathcal{V}(\mathcal{F})$ zu c^* minimiert wird:

$$\begin{aligned} \min \quad & r(\mathbf{c}) = \max_{v \in \mathcal{V}(\mathcal{F})} \|\boldsymbol{v} - \mathbf{c}\| \\ \text{s.d.} \quad & \mathbf{c} \in \mathbb{R}^3 \end{aligned}$$

Den gesuchten Kugelradius erhalten wir somit als:

$$r^* := r(\mathbf{c}^*) .$$

Die Funktion $r(\mathbf{c}) : \mathbb{R}^3 \to \mathbb{R}$ ist offensichtlich nicht differenzierbar, jedoch konvex, wie die folgende Betrachtung zeigt:
Für alle $v \in \mathcal{V}(\mathcal{F})$ und $\lambda \in [0, 1]$ gilt:

$$\begin{aligned} \|\boldsymbol{v} - (1-\lambda)\mathbf{c}_1 - \lambda\mathbf{c}_2\| &= \|(1-\lambda)(\boldsymbol{v} - \mathbf{c}_1) + \lambda(\boldsymbol{v} - \mathbf{c}_2)\| \\ &\leq (1-\lambda)\|\boldsymbol{v} - \mathbf{c}_1\| + \lambda\|\boldsymbol{v} - \mathbf{c}_2\| . \end{aligned}$$

Wir erhalten somit:

$$\begin{aligned} r\big((1-\lambda)\mathbf{c}_1 + \lambda\mathbf{c}_2\big) &= \max_{v \in \mathcal{V}(\mathcal{F})} \|\boldsymbol{v} - (1-\lambda)\mathbf{c}_1 - \lambda\mathbf{c}_2\| \\ &\leq \max_{v \in \mathcal{V}(\mathcal{F})} (1-\lambda)\|\boldsymbol{v} - \mathbf{c}_1\| + \lambda\|\boldsymbol{v} - \mathbf{c}_2\| \\ &\leq (1-\lambda) \max_{v \in \mathcal{V}(\mathcal{F})} \|\boldsymbol{v} - \mathbf{c}_1\| + \lambda \max_{v \in \mathcal{V}(\mathcal{F})} \|\boldsymbol{v} - \mathbf{c}_2\| \\ &= (1-\lambda)r(\mathbf{c}_1) + \lambda r(\mathbf{c}_2) . \end{aligned}$$

Zwar können wir Verfahren der nichtlinearen konvexen Optimierung wie den *Downhill-Simplex-Algorithmus* [NM65] einsetzen, um das Problem ohne Berechnung von Gradienten zu lösen, doch existieren effiziente kombinatorische Verfahren, die in der Praxis deutlich bessere Laufzeitergebnisse liefern.

Der randomisierte Algorithmus von WELZL [Wel91] löst das Problem durch inkrementelle Vorgehensweise in erwarteter Linearzeit, so daß eine Berechnung der konvexen Hülle von $\mathcal{F}$ als Beschleunigungsansatz keinen Sinn mehr macht.

Der Algorithmus von WELZL

Wir werden nun den Algorithmus von WELZL [Wel91] vorstellen, der die kleinste einschließende Kugel einer endlichen Menge von Punkten in erwarteter Linearzeit berechnet. Die Idee des Algorithmus' basiert auf einem Verfahren von SEIDEL [Sei90] zur Lösung linearer Programme. Neben dem WELZL-Algorithmus existiert ein schwieriger zu implementierender, deterministischer Linearzeitalgorithmus von MEGGIDO [Meg83], der aufgrund der höheren Konstanten jedoch von geringerem praktischen Interesse ist.

Das nun vorzustellende Verfahren arbeitet inkrementell. Es startet mit einer leeren Menge von Punkten und nimmt sukzessive die zu umhüllenden Punkte hinzu. Dabei muß die kleinste umschließende Kugel für die bisher betrachtete Punktmenge eventuell so modifiziert werden, daß sie auch für die neue Punktmenge optimal ist.

Sei $P = \{p_1, \ldots, p_n\}$ die einzuschließende Punktmenge und $S^*(P)$ die volumenminimale Kugel mit der Eigenschaft $P \subseteq S^*(P)$. Des weiteren bezeichne $S^*(P, R)$ die kleinste einschließende Kugel der Punktmenge P, für die gilt, daß alle Punkte aus R auf dem Rand der Kugel liegen. Aufgrund der Beziehung $S^*(P, \emptyset) = S^*(P)$ können wir mit einem Algorithmus zur Bestimmung von $S^*(P, R)$ unser ursprüngliches Problem lösen. Das folgende Lemma stellt die Grundlage dieses Algorithmus' dar:

Lemma 3.5. *Seien A und R endliche Punktmengen des $\mathbb{R}^3$, A sei nichtleer und $p \in A$. Dann gilt:*

(i) Falls eine A umhüllende Kugel existiert, für die gilt, daß alle Punkte aus R auf dem Rand liegen, so ist $S^*(A, R)$ eindeutig bestimmt.

(ii) Falls p nicht in $S^*(A \setminus \{p\}, R)$ liegt, dann muß p auf dem Rand von $S^*(A, R)$ liegen, sofern $S^*(A, R)$ existiert. Es gilt also:

$$S^*(A, R) = S^*(A \setminus \{p\}, R \cup \{p\}) .$$

(iii) Falls $S^*(A, R)$ existiert, dann gibt es höchstens $\max\{0, 4 - |R|\}$ Punkte in A, für die gilt $p \notin S^*(A \setminus \{p\}, R)$.

Aussage (ii) eröffnet ein einfaches, rekursives Verfahren zur Berechnung von $S^*(A, R)$, welches durch Algorithmus 12 dargestellt ist.

Falls $A = \emptyset$ gilt, können wir $S^*(\emptyset)$ unmittelbar und in konstanter Zeit ermitteln, indem wir eine Kugel durch die Randpunkte aus R legen. Auch der Fall $|R| = 4$ kann sofort beantwortet werden, da vier vorgegebene Randpunkte eine Kugel eindeutig bestimmen. In allen anderen Fällen wählen wir einen zufälligen Punkt $p \in A$ und berechnen $S = S^*(A \setminus \{p\}, R)$. Liegt p innerhalb von S, so ist S die kleinste A umschließende

Algorithmus 12 Ein Algorithmus zur Bestimmung der kleinsten einschließenden Kugel bei vorgegebenen Randpunkten.

Eingabe: Die zu umhüllende Punktmenge A und die Randpunktmenge R (Voraussetzung: $S^*(A, R)$ existiert).
Ausgabe: Die A einschließende Kugel minimalen Volumens, deren Randpunkte die Menge R beinhalten: $S^*(A, R)$.

```
MINSPHERE(A, R)
(1)    if P = ∅ or |R| = 4
(2)       (* Problem kann unmittelbar gelöst werden *)
(3)       return S*(∅, R)
(4)    else
(5)       p ← RANDOMMEMBER(A)
(6)       S ← MINSPHERE(P \ {p}, R)
(7)       if p ∉ S
(8)          S ← MINSPHERE(P \ {p}, R ∪ {p})
(9)       return S
```

Kugel mit der Eigenschaft, daß jeder Punkt aus R auf dem Rand von S liegt. Andernfalls folgt aus Aussage (ii), daß sich p auf dem Rand von $S^*(A, R)$ befinden muß. Es gilt daher:

$$S^*(A, R) = S^*(A \setminus \{p\}, R \cup \{p\}) .$$

Bisher haben wir eine wichtige Voraussetzung für die Anwendung des Lemmas ignoriert, nämlich die Frage, ob die gesuchten Kugeln mit den vorgegebenen Randpunkten überhaupt existieren. Diese Eigenschaft ist jedoch garantiert, falls wir Algorithmus 12 zur Bestimmung der kleinsten P umschließenden Kugel einsetzen, d.h. wenn wir $A = P$ und $R = \emptyset$ als Startwerte übergeben. Hinter dieser Behauptung steht ein „globales“ Argument. Aus der Tatsache, daß das ursprüngliche Problem (Bestimmung von $S^*(P, \emptyset)$) eine Lösung besitzt, folgt, daß dies auch für jedes von dem Algorithmus aufgeworfene Teilproblem $S^*(A, R)$ gelten muß.

Im Rahmen der Analyse des Algorithmus wollen wir die Zahl der Tests „$p \in S^*(A \setminus \{p\}, R)$“ in Zeile 7 zählen. Die tatsächliche Laufzeit ist lediglich ein konstantes Vielfaches dieser Zahl. Sei $T_j(n)$ dieses Kostenmaß für einen Aufruf von MINSPHERE(A, R) mit $|P| = n$ und $|R| = 4 - j$. Offensichtlich erhält man für $j = 0$ bzw. $n = 0$: $T_0(n) = 0$ und $T_j(0) = 0$. Falls nun $j > 0$ und $n > 0$ gilt, so erfolgt ein Aufruf von MINSPHERE$(A \setminus \{p\}, R)$, wobei $p \in A$ zufällig gewählt wurde. Falls p nicht in $S^*(A \setminus \{p\}, R)$ liegt, so wird die Routine MINSPHERE mit den Argumenten $A \setminus \{p\}$ und $R \cup \{p\}$ aufgerufen. Nach (iii) wissen wir, daß es höchstens $4 - |R| = j$ Punkte in A geben kann, die diesen letzten rekursiven Aufruf auslösen können, so daß die Wahrscheinlichkeit für diesen Testausgang

$$\mathrm{Prob}\big(S^*(A, R) \neq S^*(A \setminus \{p\}, R)\big) \leq \frac{j}{n}$$

beträgt. Wir erhalten somit die folgende Kostenfunktion:

$$T_j(n) = T_j(n-1) + 1 + \frac{j}{n} T_{j-1}(n-1) \, .$$

Die Auflösung der Rekursionsgleichung für $1 \leq j \leq 3$ zeigt:

$$T_j(n) = O(n) \quad 1 \leq j \leq 3,$$

woraus wir ableiten können, daß die kleinste umschließende Kugel für die Punktmenge P in erwarteter Linearzeit gefunden werden kann.

Der einschließende Quader minimalen Volumens

Wir wollen nun die Flächenmenge $\mathcal{F}$ in einen Quader $B(\mathbf{c}, \mathbf{D}, \overline{\mathbf{d}})$ derart einschließen, daß dessen Volumen minimiert wird. Die Matrix $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]$ mit $\|\mathbf{d}_i\| = 1$, $1 \leq i \leq 3$, gibt dabei die paarweise aufeinander senkrecht stehenden Richtungen der Quaderachsen an. Der Vektor $\mathbf{c}$ entspricht dem Quadermittelpunkt und der Vektor $\overline{\mathbf{d}} = (\overline{d}_1, \overline{d}_2, \overline{d}_3)$ dem maximalen Abstand eines Punktes von $\mathcal{F}$ zu Punkt c in Richtung von $\mathbf{d}_i$, $1 \leq i \leq 3$. Das Volumen des Quaders B ergibt sich daher als

$$V(B) = 8 \prod_{i=1}^{3} \overline{d}_i \, , \tag{3.12}$$

$$\text{mit} \quad \overline{d}_i = \frac{1}{2}\big(\max_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i^T \boldsymbol{v} - \min_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i^T \boldsymbol{v} \big), \quad 1 \leq i \leq 3 \, . \tag{3.13}$$

Somit erhalten wir als Ortsvektor des Quadermittelpunkts:

$$\mathbf{c} \;=\; \frac{1}{2} \sum_{i=1}^{3} \big(\min_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i^T \boldsymbol{v} + \max_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i^T \boldsymbol{v} \big) \mathbf{d}_i$$

Die Matrix $\mathbf{D}$ ist eine Rotationsmatrix, deren Spaltenvektoren $\mathbf{d}_i$ sich durch sukzessive Drehung des i-ten Einheitsvektors um die Koordinatenachsen ergeben ($1 \leq i \leq 3$). Bezeichnen wir die Rotationswinkel mit (α, β, γ) für die Drehung um $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$, so gilt:

$$\mathbf{D} = \mathbf{R}(\alpha, \beta, \gamma) \, .$$

Wir können somit ein nichtlineares und nichtkonvexes Optimierungsproblem in den Parametern α, β, γ formulieren, das den einschließenden Quader minimalen Volumens $B^*(\mathcal{F}) = B(D^*, c^*, \overline{\mathbf{d}}^*)$ berechnet:

$$\begin{aligned} \min \quad & \prod_{i=1}^{3} \big(\max_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i(\alpha, \beta, \gamma)^T \boldsymbol{v} - \min_{v \in \mathcal{V}(\mathcal{F})} \mathbf{d}_i(\alpha, \beta, \gamma)^T \boldsymbol{v} \big) \\ \text{s.d.} \quad & \alpha, \beta, \gamma \in [0, 2\pi] \end{aligned}$$

Neben den zeitaufwendigen Optimierungsmethoden für solche Probleme (z.B. *simulated annealing*) erscheinen auch Verfahren der nichtlinearen, konvexen Optimierung wie beispielsweise der *Downhill-Simplex-Algorithmus* [NM65] lediglich eingeschränkt praktikabel, da die numerische Minimumsuche auf mehrere Startwerte angewendet werden muß und eine optimale Lösung nicht immer garantiert ist.

Ähnlich wie im Fall der umhüllenden Kugel minimalen Volumens existiert ein kombinatorischer Algorithmus von O'ROURKE [O'R85], der den einschließenden Quader minimalen Volumens in Zeit $O(n^3)$ berechnet. Dabei bezeichnet n die Zahl der Eckpunkte der konvexen Hülle von $\mathcal{F}$.

Der Algorithmus von O'ROURKE

Da der Algorithmus aufgrund seiner Laufzeit nur von geringem praktischen Interesse ist, wollen wir in diesem Abschnitt lediglich eine Skizze der Funktionsweise des Verfahrens liefern. Eine umfassende Darstellung findet sich in [O'R85].

Zunächst muß die konvexe Hülle $CH(V(\mathcal{F}))$ der Punktmenge $\mathcal{V}(\mathcal{F})$ gebildet werden, so daß die Aufgabe darin besteht, einen konvexen Polyeder P mit n Eckpunkten in einen Quader minimalen Volumens einzuhüllen.

Definition 3.4. Eine Seitenfläche f eines einschließenden Quaders liegt an einer Kante e des zu umschließenden Polyeders an, falls $e \subseteq f$ gilt. □

Die folgende Erkenntnis bildet den Kern des Verfahrens.

Satz 3.6. *Ein Quader minimalen Volumens, der einen konvexen Polyeder einschließt, besitzt mindestens zwei adjazente Flächen, die an Kanten des Polyeders anliegen.*

Der Algorithmus zählt nun alle Kantenpaare auf und benutzt die in Satz 3.6 garantierte Eigenschaft, um für dieses Kantenpaar zwei anliegende, adjazente Quaderflächen zu finden, die einen einhüllenden Quader minimalen Volumens induzieren.

Fixiert man zwei Kanten an adjazenten Quaderflächen, so kann gezeigt werden, daß für die Orientierung des Quaders lediglich ein Freiheitsgrad besteht.

Lemma 3.7. *Seien $\mathbf{d}_1$ und $\mathbf{d}_2$ die Normalen der beiden adjazenten Flächen, die an Kanten e_1 und e_2 des Polyeders anliegen und sei $\mathbf{d}_3$ die auf $\mathbf{d}_1$ und $\mathbf{d}_2$ senkrecht stehende Normale, dann gilt, daß $\mathbf{d}_1$ die Normalen $\mathbf{d}_2$ und $\mathbf{d}_3$ eindeutig bestimmt.*

Da die durch $\mathbf{d}_1$ definierte Quaderfläche an der Kante e_1 anliegt, folgt, daß $\mathbf{d}_1$ durch einen Winkel $\vartheta_1 \in [\underline{\vartheta}_1, \overline{\vartheta}_1]$, mit $\overline{\vartheta}_1 - \underline{\vartheta}_1 = \pi$ festgelegt werden kann. Die Oberflächenelemente, die die Quaderflächen bei Variation dieses Parameters, der sogenannten *Caliper-Drehung*, berühren, bilden den entsprechend benannten *Caliper-Pfad*.

Der Autor verwendet eine *duale Repräsentation* des konvexen Polyeders P, die sogenannte *Gaußkugel* bezüglich P, $S_G(P)$, um die Caliper-Drehung zu verfolgen. Die Gaußkugel unterteilt die Oberfläche der ursprungszentrierten Einheitskugel in *konvexe Regionen* $\mathcal{R}(v)$, wobei v ein Eckpunkt von P ist. Diese Aufteilung erhält man durch die folgende Konstruktion. Ist $\mathbf{n}$ ein Einheitsvektor aus dem Urspung, dessen Ende in $\mathcal{R}(v)$ liegt, dann stellt die Ebene durch v mit Normalenvektor $\mathbf{n}$ eine unterstützende

Ebene des Polyeders dar. Somit entspricht jeder Eckpunkt v von P einer Region $\mathcal{R}(v)$ auf $S_G(P)$ und jede Fläche f von P einem „Eckpunkt" auf $S_G(P)$, gegeben durch die Normale der unterstützenden Ebene für f. Jede Kante von P findet ihre Entsprechung in einem Bogen eines Großkreises von $S_G(p)$, der die Normalen aller unterstützenden Ebenen von P repräsentiert, welche diese Kante enthalten. Eine Polyederdatenstruktur mit Kanten- und Flächenadjazenzinformationen gestattet die Konstruktion der Gaußkugel in Linearzeit.

Jeder P umschließende Quader kann somit durch drei Paare einander gegenüberliegender Punkte auf $S_G(P)$ dargestellt werden. Lemma 3.7 erklärt, daß lediglich eine Achsenorientierung bei gegebenem Kantenpaar offen bleibt, so daß die Punktepaare sich auf bestimmten Pfaden über die Gaußkugel bewegen. Für jedes Kantenpaar (e_1, e_2) müssen wir also den Wert ϑ_1^* bestimmen, der das Volumen des Quaders mit zwei Seitenflächen adjazent zu e_1 und e_2 minimiert. Das Tripel $(e_1^*, e_2^*, \vartheta_1^*)$, das den Quader kleinsten Volumens definiert, stellt die Lösung unseres Optimierungsproblems dar.

Lemma 3.8. *Wenn $\mathbf{d}_1$ innerhalb des erlaubten Bereichs variiert, laufen die mit $\mathbf{d}_1$ und $\mathbf{d}_2$ assoziierten Punkte entlang von Großkreisbögen der Gaußkugel und $\mathbf{d}_3$ folgt einem gekrümmten konvexen Pfad auf $S_G(P)$.*

Für einen festen Wert von ϑ_1 sind alle Richtungsvektoren des Quaders fixiert und die beiden an e_1 und e_2 anliegenden Flächen eindeutig bestimmt. Die verbleibenden vier Flächen können durch Berechnung der Extrempunkte in der jeweiligen Achsenrichtung in Zeit $O(n)$ berechnet werden.

Der Startquader ist bestimmt durch den niedrigsten erlaubten Wert $\underline{\vartheta}_1$ für ϑ_1. Anschließend muß der nächste Wert $\vartheta_1 + \Delta\vartheta_1$ innerhalb des Intervalls $[\underline{\vartheta}_1, \overline{\vartheta}_1]$ ermittelt werden. Diesen Wert erhalten wir, indem wir den Winkel ϑ_1 kontinuierlich erhöhen, bis zum ersten Mal einer der Seitenflächenkontakte wechselt. Dies ist genau dann der Fall, wenn ein Pfad für $-\mathbf{d}_1, -\mathbf{d}_2$ oder $\pm\mathbf{d}_3$ eine Kante zwischen zwei Regionen $\mathcal{R}(v_1)$ und $\mathcal{R}(v_2)$ der Gaußkugel überschreitet. Für alle Pfade kann nun gezeigt werden, daß jede Kante von $S_G(P)$ höchstens zweimal geschnitten wird. Der Pfad, dessen nächster Schnittpunkt das kleinste Parameterintervall $\Delta\vartheta_1$ definiert, legt die neue kombinatorische Orientierung des Quaders fest. Die Anzahl der $\Delta\vartheta_1$-Schritte kann somit über die Anzahl der Kanten der Gaußkugel durch $O(n)$ abgeschätzt werden.

Innerhalb eines $\Delta\vartheta_1$-Intervalls bleiben die Kontakte der nicht an e_1 und e_2 anliegenden Seitenflächen unverändert. Der Quader minimalen Volumens kann innerhalb eines solchen Intervalls in konstanter Zeit ermittelt werden, so daß die volumenminimierende Orientierung innerhalb des gesamten ϑ_1-Intervalls in Linearzeit bestimmt werden kann.

Der Pseudoalgorithmus zur Berechnung des $\mathcal{F}$ einschließenden Quaders minimalen Volumens kann somit folgendermaßen formuliert werden.

Aufgrund der Laufzeit von $O(n^3)$ haben Heuristiken zur Bestimmung des $\mathcal{F}$ einschließenden Quaders eine größere Bedeutung erlangt [GLM96]. Der Aufbau einer Hüllkörperhierarchie für einen Körper mit einer Komplexität von n Flächen beansprucht unter Verwendung des Verfahrens von O'ROURKE eine asymptotische Laufzeit

Algorithmus 13 Ein Algorithmus zur Bestimmung des einschließenden Quaders minimalen Volumens.

Eingabe: Die Flächenmenge $\mathcal{F}$ bzw. die Menge der Eckpunkte $\mathcal{V}(\mathcal{F})$.
Ausgabe: Der Quader minimalen Volumens, welcher $\mathcal{F}$ umhüllt.

MINBOX($\mathcal{F}$)
(1) $P \leftarrow$ KOVEXEHÜLLE($\mathcal{V}(\mathcal{F})$)
(2) $S_G(P) \leftarrow$GAUSSKUGEL(P)
(3) **foreach** $(e_1, e_2) \in \mathcal{E}(P) \times \mathcal{E}(P)$
(4) Bestimme Zulässigkeitsbereich $\vartheta_1 \in [\underline{\vartheta}_1, \overline{\vartheta}_1]$ für ϑ_1
(5) **while** $\vartheta_1 < \overline{\vartheta_1}$
(6) Bestimme die nächsten Schnittpunkte der Pfade bezüglich $-\mathbf{d}_1, -d_2$ und $\pm\mathbf{d}_3$ mit Kanten von $S_G(P)$
(7) $\Delta\vartheta_1 \leftarrow$ kleinste Parameteränderung die zu einem Schnittpunkt führt
(8) Finde volumenminimalen Quader im Intervall $[\vartheta_1, \Delta\vartheta_1]$
(9) $\vartheta_1 \leftarrow \vartheta_1 + \Delta\vartheta_1$
(10) **return** $B(\mathbf{D}^*, \mathbf{c}^*, \overline{\mathbf{d}}^*) = B(e_1^*, e_2^*, \vartheta_1^*)$

von $O(n^3)$. In realen Anwendungen mit vielen Objekten hoher Polygonzahl ist ein solcher Rechenaufwand selbst in der Vorberechnungsphase nicht akzeptabel. Wir haben uns daher im Rahmen des SILVIA-Projekts dafür entschieden, eine Variante der Heuristik von GOTTSCHALK ET AL. [GLM96] zu implementieren.

Die Heuristik von GOTTSCHALK ET AL.

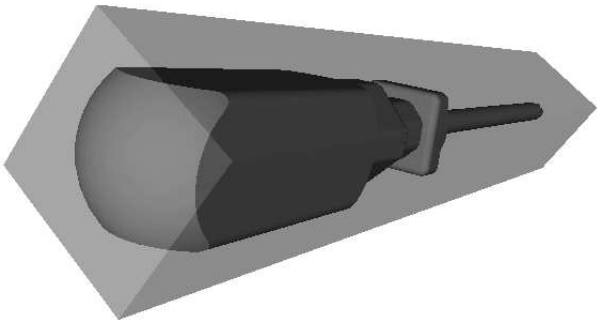
Der Vorschlag von GOTTSCHALK ET AL. besteht darin, den Quader entlang der *Hauptachsen der Flächenmenge* $\mathcal{F}$ auszurichten. Diese entsprechen den Spalten der *Trägheitsmatrix* von $\mathcal{F}$, die man erhält, indem man die Eigenvektoren der Kovarianzmatrix aller Eckpunkte ermittelt.

Um den Einfluß „innerer" Punkte auf die Orientierung des Quaders zu begrenzen, kann man sich bei der Berechnung der Kovarianzmatrix auf die konvexe Hülle der Eckpunkte $CH(\mathcal{V}(\mathcal{F}))$ beschränken. Das Beispiel des Schraubenziehergriffs in Abbildung 3.17 zeigt jedoch, daß bei gekrümmten Oberflächen, die durch viele Polygone kleiner Oberfläche approximiert werden, die Punkte der konvexen Hülle immer noch sehr dicht zusammenliegen und auf diese Weise die Wahl der Quaderachsen negativ beeinflussen können. Das Volumen des Quaders wäre geringer gewesen, wenn seine Seitenflächen nicht an die abgerundeten Kanten, sondern an die flachen Seiten des Griffs angepaßt worden wären.

Das Problem kann jedoch gelöst werden, indem nicht nur die Eckpunkte der konvexen Hülle, sondern deren gesamte Oberfläche bei der Bestimmung der Kovarianzmatrix betrachtet wird. Dazu integriert man über alle Oberflächenpunkte eines Polygons und normalisiert diesen Wert mit Hilfe der Flächeninhalte.

Trianguliert man die konvexe Hülle in n Dreiecke [Sch99], so ergibt sich bei dieser

Abbildung 3.17 Gekrümmte Flächen können nach ihrer Triangulierung die Wahl der Quaderachsen negativ beeinflussen, da die Eckpunkte sehr dicht nebeneinander liegen.



Vorgehensweise für den „Mittelpunkt“ der Flächenmenge:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{a^i}\int_0^1\int_0^{1-t} \boldsymbol{x}^i \, d\lambda_1 \, d\lambda_2\right) = \frac{1}{6n}\sum_{i=1}^{n}\frac{1}{a^i}\left(\boldsymbol{v}_1^i + \boldsymbol{v}_2^i + \boldsymbol{v}_3^i\right) ,$$

wobei $\boldsymbol{x}^i = \boldsymbol{v}_1^i + \lambda_1(\boldsymbol{v}_2^i - \boldsymbol{v}_1^i) + \lambda_2(\boldsymbol{v}_3^i - \boldsymbol{v}_1^i)$, $\lambda_1, \lambda_2 \in [0,1]$ der Darstellung eines Oberflächenpunktes $\boldsymbol{x}^i$ als Konvexkombination der Dreieckseckpunkte $\boldsymbol{v}_1^i$, $\boldsymbol{v}_2^i$ und $\boldsymbol{v}_3^i$ entspricht. Das Skalar $a_i = \frac{1}{2}\|(\boldsymbol{v}_2^i - \boldsymbol{v}_1^i) \times (\boldsymbol{v}_3^i - \boldsymbol{v}_1^i)\|$ ist der Flächeninhalt des Dreiecks i, $i \in \{1, \ldots, n\}$. Die Einträge c_{jk} der Kovarianzmatrix $\mathbf{C}$ ergeben sich somit als:

$$\begin{aligned} c_{jk} \;=\; & \frac{1}{24n}\sum_{i=1}^{n} a^i \Big[(\overline{v}_{1j}^i + \overline{v}_{2j}^i + \overline{v}_{3j}^i)(\overline{v}_{1k}^i + \overline{v}_{2k}^i + \overline{v}_{3k}^i) \\ & + \overline{v}_{1j}^i \overline{v}_{1k}^i + \overline{v}_{2j}^i \overline{v}_{2k}^i + \overline{v}_{3j}^i \overline{v}_{3k}^i\Big] , \quad 1 \le j,k \le 3 , \end{aligned}$$

mit $\overline{\boldsymbol{v}}_l^i = (\overline{v}_{l1}^i, \overline{v}_{l2}^i, \overline{v}_{l3}^i)^T = \boldsymbol{v}_l^i - \boldsymbol{\mu}$, $1 \le l \le 3$.
Die gesuchten Achsenrichtungen d_1, d_2 und d_3 erhalten wir, indem wir die normierten Eigenvektoren der Matrix $\mathbf{C}$ ermitteln.

Die Fixed Directions Hulls (FDH_n)

Dieser Hüllkörpertyp ist in der Literatur [KZ97], [Kon98] als eine Sammlung von Halbebenen definiert, die die Flächenmenge $\mathcal{F}$ begrenzen und deren Schnitt mit jeder unterstützenden Ebene von $\mathcal{F}$ einen gemeinsamen Punkt besitzt. Die $\mathsf{FDH}_n(\mathcal{F})$ ist bestimmt durch eine Matrix $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_n]$, die die normierten Richtungen der begrenzenden Halbebenen vorgibt und n Intervalle $[\,\overline{d}_{2i-1}, \overline{d}_{2i}]$ mit $\mathbf{d}_{2i} = -\mathbf{d}_{2i-1}$, die die Ausdehnung der FDH_n entlang der n Richtungen $\mathbf{d}_{2i-1}$, $i = 1, \ldots, n$ beschreiben. Vom Volumen einer FDH_n zu sprechen macht daher nur Sinn, wenn wir, unserer Intuition folgend, das

Volumen als das Schnittvolumen der FDH_n-Halbebenen definieren:

$$V(\mathrm{FDH}_n) := V\left(\bigcap_{i=1}^{n} \mathcal{H}(\mathbf{d}_i, \overline{\mathbf{d}})\right) ,$$

wobei $\mathcal{H}(\mathbf{d}_i, \overline{\mathbf{d}})$ die Halbebene $\mathcal{H} = \left\{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{d}_i^T x \leq \overline{d}_i\right\}$, $1 \leq i \leq n$ bezeichnet.
Das Gütemaß ist somit wohldefiniert, falls der Schnitt der n Halbebenen einen nichtleeren konvexen Polyeder definiert. Diese Voraussetzung kann durch eine geeignete Wahl der Richtungsvektoren sichergestellt werden. In [KZ97] und [Kon98] werden die sogenannten FDH_{14}-Hüllkörper verwendet, die durch die folgende Matrix $\mathbf{D}$ definiert sind:

$$\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & -\nu & \nu & \nu & -\nu & -\nu & \nu & -\nu & \nu \\ 0 & 0 & 1 & -1 & 0 & 0 & -\nu & \nu & -\nu & \nu & \nu & -\nu & -\nu & \nu \\ 0 & 0 & 0 & 0 & 1 & -1 & -\nu & \nu & -\nu & \nu & -\nu & \nu & \nu & -\nu \end{pmatrix}^T ,$$

mit $\nu := 1/\sqrt{3}$.

Im Gegensatz zur Situation bei Kugeln und Quadern ist im Fall der FDH_n keine explizite Volumenminimierung erforderlich. Der Vektor $\overline{\mathbf{d}} = [\overline{d}_1, \ldots, \overline{d}_n]$ ist gerade so definiert, daß die Ausdehnung der FDH_n entlang den vorgegebenen Richtungen minimal ist. Da diese Orientierungen fest sind, wird das minimale Volumen eines $\mathcal{F}$ umfassenden Halbraumschnitts durch die $\mathrm{FDH}_n(\mathbf{D}, \overline{\mathbf{d}})$ angenommen. Hinsichtlich der Notation erweist es sich oft als günstiger, für die Aufzählung der Richtungsvektoren auf eine Mengenschreibweise zurückzugreifen. Dazu bezeichnen wir die Menge der Richtungsvektoren, die in der Matrix $\mathbf{D}$ als Spaltenvektoren kodiert sind, mit $D := \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$.

3.6 Aufbau der Hüllkörperhierarchie

3.6.1 Die hierarchische Partitionierung der Flächenpaarmenge

Das von uns vorgestellte Branch-and-Bound-Verfahren zur Abstandsberechnung zweier Objekte minimiert die Abstandsfunktion über die Menge aller Flächenpaare $X = \mathcal{F}_1 \times \mathcal{F}_2$ der beiden Körper. Die Effizienz dieses Algorithmus hängt dabei sehr stark von der Güte der berechneten unteren Schranken und somit von der Approximationsqualität der Hüllkörper ab. Abgesehen davon hat auch die Gestalt des Suchbaums, d.h. die hierarchische Partitionierung der Alternativenmenge X, einen wesentlichen Einfluß auf den Berechnungsaufwand. Die Hierarchie ergibt sich dabei aus einer zunehmend feineren Partitionierung von X auf den einzelnen Hierarchieebenen und endet im Extremfall auf einer Ebene, die eine vollständige Zerlegung von X in einzelne Flächenpaare darstellt. Dabei ist jedoch zu beachten, daß günstige Unterteilungen von X bestimmte geometrische Eigenschaften erfüllen. Anschaulich betrachtet, verfolgen wir mit der Partitionierung von X das Ziel, Paare von Randteilen, die einen größeren Abstand zueinander besitzen als andere, als Lieferant des nahesten Punktepaares ausschließen zu

können. Es macht daher keinen Sinn die Menge der Flächenpaare X ohne Beachtung räumlicher Korrelationen der Flächen aus $\mathcal{F}_1$ bzw. $\mathcal{F}_2$ zu unterteilen. Statt dessen sollten alle Teilmengen der Partition aus zusammenhängenden Flächenkomplexen bestehen.

Theoretisch dürfte eine geeignete Partition erst zum Zeitpunkt der Abstandsberechnung der beiden Körper erzeugt werden, da das beteiligte Objektpaar nicht a priori bekannt ist und man nur aus der aktuellen Lage der beiden Körper zueinander eine optimale Unterteilung der Flächenpaare ermitteln kann. Da ein solcher Hierarchieaufbau in der eigentlichen Simulationsphase zu teuer ist, kann man versuchen, den Aufwand für die *online-Konstruktion* der Hierarchie zu reduzieren. Dazu ermittelt man außerhalb der eigentlichen Simulationsphase, d.h. offline für jeden Körper, eine hierarchische Partitionierung seines Körperrandes. Zum Zeitpunkt der Abstandsberechnung erhält man eine Partition von X, indem man eine Ebene der hierarchischen Raumunterteilung von Körper 1 und eine von Körper 2 miteinander verknüpft. Sei $N_1 = \{\mathcal{F}_{11}, \ldots, \mathcal{F}_{1n}\}$ bzw. $N_2 = \{\mathcal{F}_{21}, \ldots, \mathcal{F}_{2m}\}$ eine Partition von $\mathcal{F}_1$ bzw. $\mathcal{F}_2$, so definieren diese die folgende Partition N von X:

$$N = N_1 \times N_2 = \left\{ X_{ij} \,\middle|\, X_{ij} = \mathcal{F}_{1i} \times \mathcal{F}_{2j},\ 1 \leq i \leq n,\ 1 \leq j \leq m \right\}.$$

Ersetzt man nun beispielsweise die Menge N_1 durch ihre feinere Unterteilung auf der nächsten Hierarchiestufe, N_1', so erhält man eine Partitionierung $N' = N_1' \times N_2$ von N mit größerer Granularität.

Im folgenden können wir uns also auf die Fragen beschränken, wie der Rand eines einzelnen Körpers sinnvoll partitioniert werden kann und welchen Einfluß die zu berechnenden Hüllkörper auf diese Aufteilung haben.

3.6.2 Homogene und heterogene Hüllkörperhierarchien

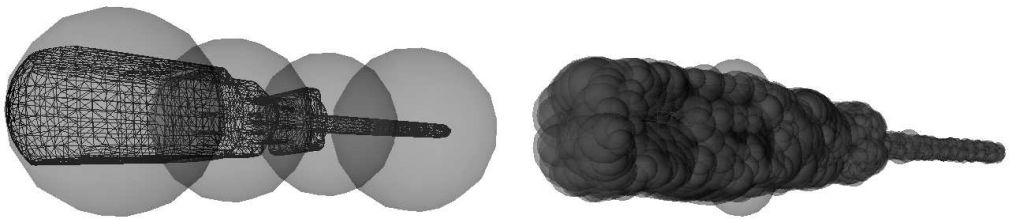
Entsprechend der gerade dargestellten Überlegungen besteht unsere Aufgabe darin, innerhalb der Vorberechnungsphase eine hierarchische Partitionierung des durch die Flächenmenge $\mathcal{F}$ beschriebenen Körperrandes zu bestimmen. Für jede entstehende Teilmenge von $\mathcal{F}$ muß ein Hüllkörper ermittelt werden, der die schnelle Berechnung einer unteren Schranke für den Abstand zu einer anderen Flächenmenge erlaubt. Eine solche Konstruktion bezeichnet man als *Hüllkörperhierarchie*.

Definition 3.5 (Hüllkörperhierarchie). Eine *Hüllkörperhierarchie* ist eine disjunkte und von Hierarchiestufe zu Hierarchiestufe verfeinerte Aufteilung des Randes des zu approximierenden Körpers, wobei jede Teilmenge der Partition durch einen Hüllkörper überdeckt ist. □

Abbildung 3.18 zeigt zwei Stufen der Kugelhierarchie für unser Schraubenzieherbeispiel. Während das linke Bild einer relativ groben Approximation des Körperrandes entspricht, sieht man im rechten Teil der Abbildung eine deutlich feinere Überdeckung der Objektflächenmenge.

Beim Aufbau einer Hüllkörperhierarchie verfolgen wir zwei Zielsetzungen.

Abbildung 3.18 Unterschiedlich fein granulierte Oberflächenüberdeckungen innerhalb der Hüllkörperhierarchie.



1. Die Hierarchie soll die Zeit, die zur Identifikation des nahesten Punktepaares benötigt wird, minimieren.

2. Die Zeit zum Aufbau und der Platzbedarf zur Speicherung der Hierarchie sollten in einem vernünftigen Verhältnis zur Komplexität des zu approximierenden Körpers stehen.

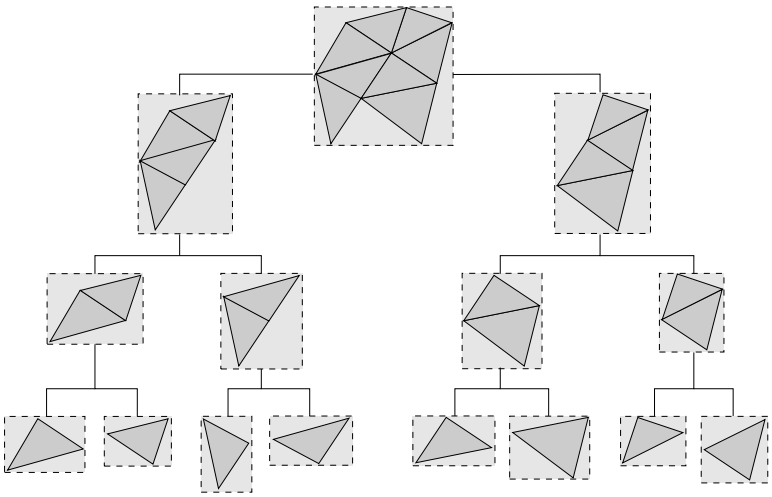
Vorgabe 2 erinnert uns daran, daß wir komplexe virtuelle Welten mit sehr vielen interagierenden Objekten simulieren wollen und deshalb den Aspekt der Praktikabilität – selbst bei Vorberechnungen – nicht aus den Augen verlieren dürfen.

Die Hüllkörperhierarchie eines Körpers $\mathcal{K}$ wollen wir durch einen Baum, den sogenannten D-*Baum*, $D(\mathcal{K})$, repräsentieren. Der Körperrand von $\mathcal{K}$ wird, wie bereits erwähnt, durch die Flächenmenge $\mathcal{F}$ beschrieben. Jeder Knoten v von $D(\mathcal{K})$ entspricht einer Teilmenge $\mathcal{F}_v$ von $\mathcal{F}$ sowie einem $\mathcal{F}_v$ überdeckenden Hüllkörper $H_v = H(\mathcal{F}_v)$. Die Wurzel r wird dabei mit der gesamten Flächenmenge $\mathcal{F}$ assoziiert. Jeder innere Knoten von $D(K)$ hat mindestens zwei und maximal $\alpha \geq 2$ Kinderknoten. Der Wert α heißt *Verzweigungsgrad* von $D(\mathcal{K})$. Die Teilmengen $\mathcal{F}_{v_1}, \ldots, \mathcal{F}_{v_k}$, $k \leq \alpha$, die den Kinderknoten von v zugeordnet sind, stellen eine Partition der Menge $\mathcal{F}_v$ dar. Auf diese Weise ist sichergestellt, daß die durch $\mathcal{F}_v$ beschriebene Randregion von $\mathcal{K}$ mit Hilfe der Hüllkörper $H_{v_1}, \ldots, H_{v_k}$ vollständig überdeckt wird. Ein Blatt w des Baumes ist dadurch gekennzeichnet, daß die ihm zugeordnete Flächenmenge $\mathcal{F}_w$ die *maximale Blattgröße* β unterschritten hat bzw. eine weitere Partitionierung von $\mathcal{F}_w$ entsprechend einer von uns vorgegebenen Aufteilungsstrategie (vgl. Abschnitt 3.6.6) nicht möglich ist. Abbildung 3.19 zeigt einen D-Baum für eine Menge von Polygonen in der Ebene ($\alpha = 2$, $\beta = 1$).

Die Hüllkörperhierarchie ist offensichtlich *zeitabhängig*. Zum Zeitpunkt der Vorberechnung ist nur die Geometrie des Körpers, nicht jedoch seine Position und Orientierung im Zeitablauf bekannt. Die Bewegung des Körpers hat jedoch Einfluß auf die Lage der Flächenmengen und somit auf die Gestalt der Hüllkörper, die sie überdecken. Diese müssen sich der Bewegung der eingeschlossenen Flächenmenge anpassen.

In dieser Arbeit wollen wir darauf verzichten, unterschiedliche Hüllkörpertypen innerhalb einer Hierarchie einzusetzen (*heterogene Hüllkörperhierarchien*). Der Aufwand

Abbildung 3.19 Der vollständig balancierte D-Baum einer Flächenmenge.



der Abstandsberechnung zwischen zwei unterschiedlichen Hüllkörpertypen steht in keinem Verhältnis zu den dabei realisierbaren Approximationsverbesserungen. Aus demselben Grund setzen wir voraus, daß diese *homogenen Hierarchien* für alle Körper der Simulation auf dem gleichen Hüllkörpertyp beruhen.

Bevor wir Verfahren zum Aufbau einer Hüllkörperhierarchie vorstellen, wollen wir den Einfluß der Hierarchie auf die Laufzeitkosten unseres Branch-and-Bound-Verfahrens beleuchten.

3.6.3 Die Kostenfunktion der Hierarchietraversierung

Durch die Homogenität der Hierarchien umgehen wir das Problem, Abstandsberechnungen zwischen unterschiedlichen Typen von Hüllkörpern durchführen zu müssen und können die Zeit für die entsprechenden Berechnungen als für alle Hüllkörperpaare identisch annehmen. Entsprechend unserer Zielsetzung, die Zeit zur Bestimmung des nahesten Punktepaares zu minimieren, werden die Laufzeitkosten unseres Abstandsberechnungsverfahrens, wie wir es in Abschnitt 3.3.4 skizziert haben, durch die folgende Kostenfunktion beschrieben:

$$C = n_D C_D + n_B C_B + n_A C_A,$$

wobei

n_D :	die Anzahl der Hüllkörperabstandsberechnungen,
C_D :	die Kosten der Hüllkörperabstandsberechnung,
n_B :	die Anzahl der elementaren Abstandsberechnungen,
C_B :	die Kosten der elementaren Abstandsberechnung,
n_A :	die Anzahl der Hüllkörperaktualisierungen,
C_A :	die Kosten der Hüllkörperaktualisierung,

bezeichnet.

Die Kosten einer Berechnung wollen wir in der Zahl der auszuführenden Fließkommaoperationen messen. Dabei unterscheiden wir das Ziehen der Quadratwurzel (Sqrt), die Multiplikation (Mult), die Addition (Add) und die Vergleichsoperationen (Comp). Um ein skalares Kostenmaß zu erhalten, müßten wir eigentlich die Kosten dieser Elementaroperationen zueinander in Beziehung setzen. Solche relativen Kosten sind jedoch maschinenabhängig, weswegen wir in dieser Arbeit keine Gewichtung vornehmen wollen.

Die Kostenfunktion ist offensichtlich *zeitabhängig*, da sich mit der Bewegung der Körper auch die nahesten Punkte zwischen diesen verändern. Somit werden die von dem Branch-and-Bound-Verfahren betrachteten Hierarchieknoten in ihrer Zahl von Aufruf zu Aufruf variieren.

Eine isolierte Betrachtung und Optimierung der einzelnen Kostenfunktionskomponenten macht wenig Sinn, da es sich dabei um *konfliktäre Zielfunktionen* handelt. So werden beispielsweise die Kosten der Hüllkörperabstandsberechnung (C_D) durch den Einsatz von AABB-Hierarchien minimiert. Wenn die Körper sich jedoch nahe kommen, so ist die Zahl der Abstandstests (n_D) im Vergleich zu OBB-Bäumen mit ihren im allgemeinen deutlich besseren Approximationseigenschaften zu hoch. Ebenso gilt, daß die Zahl der elementaren Abstandstests (n_B) durch Hüllkörper hoher Approximationsgüte niedrig gehalten wird, während durch eine solche Hüllkörperwahl die Kosten sowie die Zahl der Hüllkörperabstandsberechnungen (C_D, n_D) steigen. Einzig die Kosten der elementaren Abstandsberechnung (C_B) sind unabhängig von der Wahl der Parameter unseres Verfahrens, weshalb wir diesen Kostenfaktor losgelöst von den übrigen Variablen in Abschnitt 3.4 minimieren konnten. Aufgrund dieser wechselseitigen Einflüsse müssen der Hüllkörpertyp, die Hierarchieparameter α, β sowie die Partitionierungsstrategie für die Flächenmengen dagegen so gewählt werden, daß die Traversierungskosten „in den meisten Fällen“ möglichst gering sind.

Die beiden folgenden Abschnitte beschrieben nun zwei prinzipielle Vorgehensweisen zum Aufbau des D-Baumes. Strategien zur konkreten Unterteilung der Flächenmengen werden im Anschluß vorgestellt.

3.6.4 Die Bottom-Up-Vorgehensweise

Der *Bottom-Up*-Ansatz zur Berechnung des D-Baumes überdeckt zunächst die Flächenmenge auf der Ebene der Blattknoten. Eine denkbare Möglichkeit wäre beispielsweise, jede einzelne Fläche einem Blattknoten zuzuteilen und durch einen Hüllkörper einzuschließen. Auf diese Weise erhält man auf Blattebene eine initiale Menge von Knoten V_0, die anschließend zu Knoten auf der nächsthöheren Hierarchieebene gruppiert werden müssen. Ein solcher Gruppierungsschritt faßt die Knoten $v_1, \ldots, v_k \in V_i$, $k \leq \alpha$, $i \geq 0$ zu ihrem Vaterknoten $v \in V_{i+1}$ zusammen. Dem Knoten v wird als Flächenmenge die Vereinigung aller Flächen seiner Kinderknoten zugeordnet: $\mathcal{F}_v = \bigcup_{j=1}^{k} \mathcal{F}_{v_j}$. Der Hüllkörper für die Flächenmenge $\mathcal{F}_v$ kann nun entweder aus den Hüllkörpern $H_{v_1}, \ldots, H_{v_k}$ bestimmt (z.B. im Fall der FDH_n) oder entsprechend unserer Überlegun-

gen in 3.5.4 über die ihm zugrundeliegende Flächenmenge berechnet werden. Der Prozeß des Baumaufbaus endet, wenn die Menge V_r nur noch einen einzigen Knoten enthält. Bei diesem handelt es sich dann um den Wurzelknoten r des D-Baumes. Eine konkrete Implementierung diese Ansatzes ist beispielsweise der sogenannte BOXTREE von BAREQUET ET AL. [BCG+96].

3.6.5 Die Top-Down-Vorgehensweise

Im Rahmen der *Top-Down*-Vorgehensweise wird die Hierarchiekonstruktion an der Wurzel r des D-Baumes begonnen. Diese bildet die initiale Knotenmenge V_0. Dem Wurzelknoten ordnet man die gesamte Flächenmenge $\mathcal{F}$ sowie einen $\mathcal{F}$ überdeckenden Hüllkörper zu. Anschließend wird der durch $\mathcal{F}$ beschriebene Rand des Körpers auf den einzelnen Hierarchieebenen partitioniert, wobei eine tiefere Hierarchieebene einer feineren Unterteilung der Flächenmengenpartition der darüberliegenden Ebene entspricht. Ein solcher Partitionierungsschritt zerlegt die mit den Knoten $v \in V_i$ assoziierten Flächenmengen in jeweils $k \leq \alpha$ disjunkte Teilmengen, wobei jede so entstehende Flächenmenge den Kinderknoten von v, nämlich $v_1, \ldots, v_k \in V_{i+1}$, zugeteilt wird. Ein Blatt von $D(\mathcal{K})$ erhält man, wenn eine solche Aufteilung des Knotens v nicht möglich ist, da

- eine maximale Hierarchietiefe erreicht ist **oder**
- die Anzahl der Flächen in v den Wert β unterschritten hat **oder**
- die Aufteilungsstrategie aus 3.6.6 keine feinere Partition von $\mathcal{F}_v$ liefert.

Die Top-Down-Vorgehensweise ist diejenige, die sowohl in der Literatur als auch in konkreten Implementierungen die größte Verbreitung gefunden hat [Zac94, GLM96, Zac98, Kon98, Eck98]. Dies liegt vor allem an der einfachen Umsetzbarkeit dieser natürlichen Vorgehensweise. Des weiteren erfordert die Bottom-Up-Konstruktion lokale Informationen über den Körperrand, wohingegen dem Aufteilungsprozeß für eine Flächenmenge im Rahmen des Top-Down-Verfahrens globale Informationen über die Flächenmenge genügen.

Die Frage, welche der beiden Strategien zu besseren Hüllkörperapproximationen führt, ist in der Literatur noch nicht betrachtet worden. Auch wir haben uns aus Gründen der Einfachheit für die Top-Down-Vorgehensweise entschieden.

3.6.6 Die Aufteilungsstrategie

Der Kern eines Top-Down-Verfahrens zum Aufbau einer Hüllkörperhierarchie ist der Prozeß der Flächenaufteilung. Dabei muß die Partition der Flächenmenge gefunden werden, die eine bestimmte Zielgröße optimiert. Die Zahl der Teilmengen dieser Partition ist durch den Verzweigungsgrad α beschränkt, weshalb wir die Bedeutung dieses Parameters etwas genauer betrachten wollen.

Die Wahl des Verzweigungsgrades

Auch bei der Wahl des Verzweigungsgrades α ist es unmöglich, allgemeingültige Aussagen über den traversierungskostenminimierenden Wert dieses Parameters zu treffen. Der Verzweigungsgrad bestimmt die Höhe der Hüllkörperhierarchie und somit die maximale Länge eines Suchpfades von der Wurzel zu den Blättern. Ist der Verzweigungsgrad hoch, so erreicht das Branch-and-Bound-Verfahren sehr schnell die Blätter der Hierarchie, an denen Abstände zwischen konkreten Oberflächenelementen gemessen werden. Auf diese Weise ergibt sich sehr früh die Möglichkeit, Suchräume auszuschließen. Gleichzeitig erhöht sich jedoch der Aufwand, der an jedem betrachteten Knoten anfällt, da bis zu α^2 Hüllkörperabstandstests durchzuführen sind.

Neben diesem *Trade-Off*, der ausschließlich die Traversierungskosten betrifft, führt ein hoher Verzweigungsgrad im allgemeinen zu komplexeren Problemen bei der Konstruktion der Hüllkörperhierarchie. Schließlich muß die Flächenmenge des Vaterknotens unter Berücksichtigung gewisser Gütemaße in α Teilmengen zerlegt werden. Je nach zugrundeliegendem Gütemaß kann dieser Unterteilungsprozeß mit steigendem Verzweigungsgrad deutlich schwieriger werden.

Es liegt daher nahe, einen *variablen Verzweigungsgrad* für die Knotenaufteilung zu erlauben. Eine umfassende Untersuchung dieser Problemstellung findet man in der Arbeit von Eckstein [Eck98]. Betrachtet man die Ergebnisse, so zeigt sich beispielsweise, daß ein Verfahren mit festem Verzweigungsgrad $\alpha = 2$ bei geeigneter Wahl der übrigen Aufteilungsparameter ähnlich gute Ergebnisse erzielen wie Verfahren, die mit variablem Verzweigungsgrad arbeiten. In unserer Implementierung haben wir daher $\alpha = 2$ gewählt, so daß $D(\mathcal{K})$ einen binären Baum darstellt.

Exakte und heuristische Aufteilungsverfahren

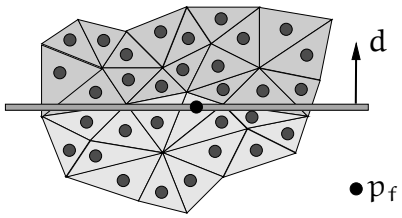
Nachdem wir den Verzweigungsgrad auf α fixiert haben, kann die Aufgabe der Flächenaufteilung nun folgendermaßen formuliert werden:

> Sei v der aufzuteilende Knoten, $\mathcal{F}_v$ die mit ihm assoziierte Flächenmenge und H_v der $\mathcal{F}_v$ überdeckende Hüllkörper. Bestimme eine Partition $\{\mathcal{F}_{v_1}, \ldots, \mathcal{F}_{v_\alpha}\}$ von $\mathcal{F}_v$, sowie Hüllkörper $H_{v_1}, \ldots H_{v_\alpha}$, so daß ein vorgegebenes Gütemaß $g : V^\alpha \to \mathbb{R}$ maximiert wird.

Betrachtet man die Problemstellung kombinatorisch, so sieht man, daß selbst für einen Verzweigungsgrad $\alpha = 2$ insgesamt $\frac{1}{2}(2^{|\mathcal{F}_v|} - 2)$ Möglichkeiten gibt, die Flächenmenge $\mathcal{F}_v$ in zwei disjunkte, nichtleere Teilmengen zu zerlegen. In der Arbeit von Eckstein [Eck98] wird ein Branch-and-Bound-Verfahren vorgestellt, das eine gütemaximierende Partitionierung von $\mathcal{F}_v$ in α Teilmengen bestimmt.

Für einige spezielle Probleme sind *komplexitätstheoretische Resultate* bekannt. Das Problem der Aufteilung einer Flächenmenge im Rahmen des Aufbaus einer Kugelhierarchie stellt beispielsweise eine Abwandlung des sogenannten *euclidean-k-center problem* [HS85] dar. Statt einer Flächenmenge ist dabei die Überdeckung einer gegebenen Punktmenge im $\mathbb{R}^d$ durch k minimale Kugeln gesucht. Das *euclidean-k-center problem* ist

Abbildung 3.20 Die Aufteilung der Flächenmenge mit Hilfe einer Trennebene.



zwar NP-vollständig für $d \geq 2$, allerdings sind Approximationsalgorithmen wie beispielsweise die Heuristik von HOCHBAUM und SCHMOYS [HS85] bekannt, die einen Approximationsfaktor von 2 erzielen.

In konkreten Implementierungen [HKM+96, GLM96, Kon98] hat sich eine sehr einfache Heuristik mit geometrischer Anschauung bewährt. Dabei werden alle Flächen mit Hilfe einer geeignet gewählten *Trennebene* (genauer: trennenden Halbebene) auf zwei Teilmengen verteilt. Das Verfahren setzt daher voraus, daß jede Fläche durch einen *Referenzpunkt*, beispielsweise durch den Mittelpunkt der kleinsten umschließenden Kugel, repräsentiert wird. Man wählt zunächst einen Richtungsvektor, der die Normale der Trennebene definiert. Anschließend positioniert man die Ebene auf der Achse, die durch den Richtungsvektor beschrieben wird. Je nachdem, auf welcher Seite der Ebene der Referenzpunkt liegt, wird die Fläche einer der beiden Teilmengen der Partition zugeordnet. Die wiederholte Anwendung dieser Technik mittels verschiedener Trennebenen erlaubt auch Partitionen der Größe 2^k, $k > 1$ zu bestimmen. Abbildung 3.20 veranschaulicht die Vorgehensweise für $k = 2$.

Wahl der trennenden Ebene

Zunächst muß die Achse festgelegt werden, auf der die Trennebene senkrecht steht. Während HELD ET AL. [HKM+96] als Alternativen die x-, y- bzw. z-Achse vorschlagen, orientieren wir uns an der Vorgehensweise von GOTTSCHALK ET AL. [GLM96]. Unsere Überlegungen in 3.5.4 zeigen, daß die Hauptachsen $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3\}$ der zu unterteilenden Flächenmenge nicht nur eine günstige Achsenwahl für den OBB-Hüllkörper darstellen, sondern auch geometrisch plausible Alternativen für die Ausrichtung der trennenden Ebene liefern. Unter diesen drei Möglichkeiten wählen wir diejenige, die die Approximationsgüte der entstehenden Hüllkörper maximiert. Dies führt uns zu unseren Überlegungen in 3.5.3 zurück. Dort haben wir bei der Suche nach einem geeigneten Gütemaß das Volumen eines Hüllkörpers dem schwieriger zu berechnenden, gerichteten Hausdorff-Abstand zwischen Hüllkörper und Flächenmenge vorgezogen. Im Rahmen des Partitionierungsproblems müssen wir die Approximationsgüten der neu entstehenden Flächenmengen zu einer skalaren Zielfunktion zusammenführen. HELD ET AL. [HKM+96] haben in diesem Zusammenhang die drei folgenden Strategien untersucht:

1. Wähle die Achse, die die Summe der Hüllkörpervolumina der Kinderknoten minimiert:
$$\min_{\mathbf{d}\in D} \sum_{j=1}^{\alpha} V\big(H_d(v_j)\big) .$$
2. Wähle die Achse, die das maximale Hüllkörpervolumen der Kinderknoten minimiert:
$$\min_{\mathbf{d}\in D} \max_{1\le j\le \alpha} V\big(H_d(v_j)\big) .$$
3. Projiziere die Referenzpunkte aller Flächen auf die drei zur Auswahl stehenden Achsen und berechne die Varianz der resultierenden Distributionen. Wähle nun die Achse mit größter Varianz:
$$\max_{\mathbf{d}\in D} \frac{1}{|\mathcal{F}_v|} \sum_{f\in\mathcal{F}_v} (\mathbf{d}^T\mathbf{p}_f - \mu)^2 , \qquad \mu := \frac{1}{|\mathcal{F}_v|} \sum_{f\in\mathcal{F}_v} \mathbf{d}^T\mathbf{p}_f .$$

Dabei bezeichnet p_f den Referenzpunkt der Fläche f, v den aufzuteilenden Knoten und $v_1, \ldots, v_\alpha$ dessen Kinderknoten. $H_d(v_j)$ ist der Hüllkörper von v_j, $1 \le j \le \alpha$, bei der Wahl der Trennebene senkrecht zu der Achsenalternative $\mathbf{d} \in D$. Dieser Überlegung liegt die Annahme zugrunde, daß der Auflagepunkt der Ebene für alle $\mathbf{d} \in D$ mit dem selben Verfahren bestimmt wird und somit nicht Gegenstand der Optimierung ist.

Vorgehensweise 3 besitzt den großen Vorteil, daß zur Auswertung der Zielfunktion eine Berechnung der Hüllkörpervolumina nicht erforderlich ist. Wird diese Information dagegen benötigt (Verfahren 1 und 2), so müssen für jede Achsenalternative neben der durch sie induzierten Flächenpartition auch die zugehörigen Hüllkörper ermittelt werden, was einen wesentlichen Einfluß auf die Laufzeit des Hierarchieaufbaus haben kann.

Nachdem wir die Normale der trennenden Ebene fixiert haben, müssen wir die optimale Lage der Ebene auf der durch die Normale verlaufenden Achse bestimmen. Da wir disjunkte, nichtleere Teilmengen von $\mathcal{F}_v$ erzeugen wollen, bleiben für $\alpha = 2$ noch $|\mathcal{F}_v| - 1$ Möglichkeiten, den Auflagepunkt der Ebene zu wählen. Will man diese Alternativen nicht alle in Betracht ziehen, so gibt es zwei natürliche Wahlmöglichkeiten für den gesuchten Punkt:

1. Wähle den Referenzpunkt der Fläche, dessen Projektion auf die bereits bestimmte Trennachse mit Richtungsvektor $\mathbf{d}$ am nächsten zu dem Mittelwert aller projizierten Referenzpunkte liegt:
$$\min_{f\in\mathcal{F}_v} \|\mathbf{d}^T\mathbf{p}_f - \mu\| .$$
2. Wähle den Median der auf die Achse projizierten Referenzpunkte.

Sowohl der Mittelwert als auch der Median können in Linearzeit bestimmt werden. Die Komplexitätsangabe bezieht sich dabei auf die Größe der zu unterteilenden Flächenmenge. Zur Berechnung des Medians setzen wir einen einfachen, randomisierten Algorithmus [CLR94] ein, der eine erwartete Laufzeit von $O(n)$ besitzt.

3.7 Abstandsberechnung zwischen Hüllkörpern

Die Abstandsberechnung zwischen Hüllkörpern erfüllt im Rahmen der Kollisionserkennung mehrere Aufgaben. Zum einen stellt der positive Abstand zweier, den Körper vollständig umschließender Primitive sowie jede untere Schranke für diesen ein *hinreichendes Kriterium für die Kollisionsfreiheit der Originalkörper* dar. Der Abstand von Hüllkörpern ist somit eine wichtige Information, um die Zahl der auf Kollision zu testenden Objektpaare zu reduzieren und auf diese Weise die Kollisionserkennung zu beschleunigen.

Möchte man die Kollisionsfreiheit eines dynamischen Systems nicht in festen Zeitschritten überprüfen, sondern den Zeitschritt so bestimmen, daß die Kollisionsfreiheit aller Objekte in diesem Zeitintervall garantiert ist, so muß man auf die in 2.6.3 vorgestellte Idee der *Kollisonszeitschranken* zurückgreifen. Es ist offensichtlich, daß man insbesondere in einer Situation, in der alle Objekte der Welt weit voneinander entfernt sind, auf eine exakte Abstandsberechnung zugunsten von unteren Schranken, wie sie von Hüllkörperabständen geliefert werden, verzichten kann.

Im Kontext unseres Branch-and-Bound-Verfahrens 3.3.4 zur exakten Abstandsberechnung liefert der Hüllkörperabstand zweier eingeschlossener Flächenmengen eine *untere Schranke für die exakte Distanz aller Flächenpaare*. Diese Schranke wird mit dem niedrigsten beobachteten Abstand zwischen Oberflächenelementen des Körpers (obere Schranke) verglichen, so daß Flächenpaare ausgeschlossen werden können, die zu weit voneinander entfernt sind, um den minimalen Abstand zwischen den Körpern zu definieren.

3.7.1 Annahmen

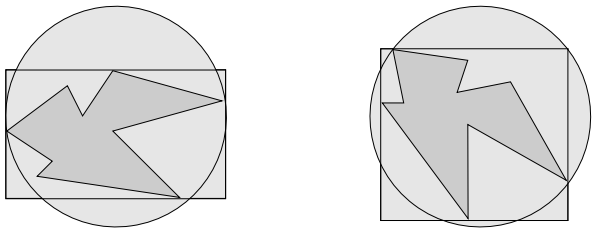
Im Rahmen der Abstandsberechnung zwischen Hüllkörpern wollen wir lediglich *Hüllkörperpaare gleichen Typs* betrachten, da die Abstandsberechnung zwischen unterschiedlichen Primitiven im allgemeinen komplex ist und es unwahrscheinlich erscheint, daß die unter Umständen besseren Approximationseigenschaften den erhöhten Rechenaufwand rechtfertigen. Dies bedeutet, daß wir uns auf *homogene Hüllkörperhierarchien* beschränken werden. Alle Körper unserer Simulation werden durch die gleichen Primitive approximiert. Es gibt innerhalb einer Hierarchie keine zwei Hüllkörper unterschiedlichen Typs.

Entsprechend unserer Abstandsdefinition aus Abschnitt 3.1 wollen wir im Fall einer Durchdringung der Hüllkörper den Abstand mit Null angeben.

3.7.2 Die Aktualisierungsproblematik

Die Bewegung eines Körpers innerhalb eines Zeitschritts verändert die Lage der von dem Hüllkörper eingeschlossenen Flächenmenge. Somit müssen auch die während der Abstandsberechnung betrachteten Hüllkörper an die transformierte Punktmenge angepaßt werden.

Abbildung 3.21 Aktualisierungsnotwendigkeit am Beispiel der Kugel bzw. Iso-Box.



Hüllkörper vor der Objektrotation ... und danach.

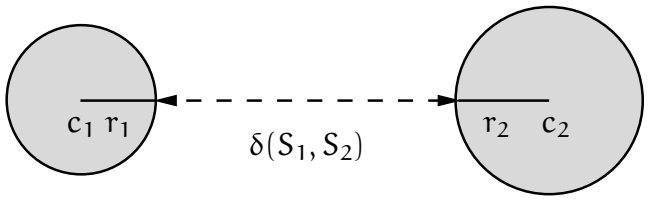
Der zur Aktualisierung notwendig Aufwand variiert zwischen den Hüllköpertypen. Er hängt im wesentlichen davon ab, ob der Hüllkörpertyp unter den in Abschnitt 2.4.1 definierten Bewegungsabbildungen abgeschlossen ist. Besitzt ein Hüllkörpertyp diese Eigenschaft, so kann ein konkreter Hüllkörper dieses Typs durch Anwendung der Körpertransformation auf einfache Weise aktualisiert werden. Beispiele hierfür sind Kugeln und Quader. Iso-Boxen und Fixed Directions Hulls erfüllen diese *Abschlußeigenschaft* nicht, so daß der Hüllkörper für die transformierte Flächenmenge entweder neu berechnet oder durch trickreichere Verfahren aktualisiert werden muß. Abbildung 3.21 verdeutlicht die Problematik am Beispiel der Kugel und der Iso-Box.

Bei der Abstandsberechnung zwischen einem Hüllkörperpaar ist es völlig ausreichend, lediglich einen der beiden beteiligten Hüllkörper zu transformieren, wie die nun folgende Betrachtung zeigt. In 2.4 haben wir gesehen, daß jede Lage, die ein starrer Körper im Rahmen einer Bewegung annimmt, durch eine Rotationsmatrix $\mathbf{R}$ und einen Verschiebungsvektor $\mathbf{t}$ dargestellt werden kann.
Seien H_i die Punktmengen, die durch die Hüllkörper beschrieben werden, und seien $\tau_i : \mathbf{x} \mapsto \mathbf{R}_i\mathbf{x} + \mathbf{t}_i$, $i = 1, 2$, die Bewegungsabbildungen der zugehörigen Körper, dann können wir den Hüllkörper H_1 in das Koordinatensystem von H_2 transformieren, indem wir beide Hüllkörper in das Weltkoordinatensystem und anschließend gemeinsam in das Körperkoordinatensystem von Objekt 2 überführen. Die Darstellung eines Hüllkörpers im Körperkoordinatensystem von $\mathcal{K}_2$ kennzeichnen wir durch das Superskript (2).

$$
\begin{aligned}
H_2^{(2)} &= \tau_2^{-1}\tau_2(H_2) = H_2\,, \\
\\
H_1^{(2)} &= \left\{\mathbf{y} \in \mathbb{R}^3 \,\middle|\, \mathbf{y} = \tau_2^{-1}\tau_1(\mathbf{x}),\ \mathbf{x} \in H_1\right\} \\
&= \left\{\mathbf{y} \in \mathbb{R}^3 \,\middle|\, \mathbf{y} = \mathbf{R}_2^T[(\mathbf{R}_1\mathbf{x} + \mathbf{t}_1) - \mathbf{t}_2],\ \mathbf{x} \in H_1\right\} \\
&= \left\{\mathbf{y} \in \mathbb{R}^3 \,\middle|\, \mathbf{y} = \mathbf{R}_2^T\mathbf{R}_1\mathbf{x} + \mathbf{R}_2^T(\mathbf{t}_1 - \mathbf{t}_2),\ \mathbf{x} \in H_1\right\} \\
&= \left\{\mathbf{y} \in \mathbb{R}^3 \,\middle|\, \mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{t},\ \mathbf{x} \in H_1\right\}\,,
\end{aligned}
$$

mit $\mathbf{R} := \mathbf{R}_2^T\mathbf{R}_1$ und $\mathbf{t} := \mathbf{R}_2^T(\mathbf{t}_1 - \mathbf{t}_2)$.
Die affine Abbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ stellt wieder eine eigentliche Bewegung dar. Ihre

Abbildung 3.22 Der Euklidische Abstand zweier Kugeln.



ausschließliche Anwendung auf den Hüllkörper des ersten Objekts transformiert das betrachtete Hüllkörperpaar in ein gemeinsames Koordinatensystem, so daß gilt:

$$\delta\big(\tau(H_1), H_2\big) = \delta\big(\tau_1(H_1), \tau_2(H_2)\big) .$$

3.7.3 Der Abstand zweier einschließender Kugeln

Aktualisierung

Die einschließende Kugel minimalen Volumens ist sowohl in ihrer Hüllkörpereigenschaft als auch in ihrer Volumenminimalität abgeschlossen unter eigentlichen Bewegungen.
Ist $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ eine eigentliche Bewegung, dann gilt:

$$S(\mathbf{c}^*, r^*) = S^*(\mathcal{F}) \iff \tau\big(S(\mathbf{c}^*, r^*)\big) = S^*\big(\tau(\mathcal{F})\big) .$$

Eine Kugel wird durch Aktualisierung des Mittelpunkts an die Bewegung der eingeschlossenen Flächenmenge angepaßt:

$$S^*\big(\tau(\mathcal{F})\big) = S\big(\tau(\mathbf{c}^*), r^*\big) .$$

Als Aktualisierungsaufwand erhalten wir für eine Kugel S:

$$C_A(S) = 9\,\mathrm{Mult} + 6\,\mathrm{Add} .$$

Abstandsberechnung

Der Abstand zweier Kugeln wird, wie in Abbildung 3.22 dargestellt, anhand des Abstands ihrer Mittelpunkte ermittelt. Ist dieser größer als die Summe der Radien, so erhalten wir den Kugelabstand, indem wir den Abstand der Mittelpunkte um die Länge der Radien verkürzen. Andernfalls liegt eine Durchdringung der Kugeln vor, weshalb wir den Abstand mit Null angeben. Für den Abstand zweier Kugeln $S_i = S(\mathbf{c}_i, r_i)$, $i = 1, 2$, erhalten wir somit:

$$\delta(S_1, S_2) = \begin{cases} \sqrt{(\mathbf{c}_1 - \mathbf{c}_2)^2} - (r_1 + r_2) & \text{falls } \sqrt{(\mathbf{c}_1 - \mathbf{c}_2)^2} > (r_1 + r_2) ; \\ 0 & \text{sonst .} \end{cases}$$

Die Kosten der Berechnung des quadratischen Abstands zweier Kugeln S_1, S_2 betragen somit:

$$C_D(S_1, S_2) = 1\,\mathrm{Sqrt} + 4\,\mathrm{Mult} + 7\,\mathrm{Add} + 1\,\mathrm{Comp}\,.$$

3.7.4 Der Abstand zweier einschließender Quader

Aktualisierung

Auch der einschließende Quader minimalen Volumens ist in seiner Eigenschaft als volumenminimaler Hüllkörper der Flächenmenge $\mathcal{F}$ abgeschlossen unter eigentlichen Bewegungen. Ist $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ eine eigentliche Bewegung, so gilt:

$$\mathrm{OBB}(\mathbf{c}^*, \mathbf{D}^*, \overline{\mathbf{d}}^*) = \mathrm{OBB}^*(\mathcal{F}) \quad \Longleftrightarrow \quad \tau\big(\mathrm{OBB}(\mathbf{c}^*, \mathbf{D}^*, \overline{\mathbf{d}}^*)\big) = \mathrm{OBB}^*\big(\tau(\mathcal{F})\big)\,.$$

Den transformierten Quader erhält man durch Aktualisierung des Mittelpunkts c^* sowie der Achsenrichtungen $\mathbf{d}_1^*, \mathbf{d}_2^*$ und $\mathbf{d}_3^*$, welche durch die Matrix $\mathbf{D}^*$ gegeben sind. Somit gilt:

$$\mathrm{OBB}\big(\tau(\mathcal{F})\big) = \tau\big(\mathrm{OBB}(\mathbf{c}^*, \mathbf{D}^*, \overline{\mathbf{d}}^*)\big) = \mathrm{OBB}\big(\tau(\mathbf{c}^*), \mathbf{R}\mathbf{D}^*, \overline{\mathbf{d}}^*\big)\,.$$

Als Aktualisierungsaufwand erhalten wir für den Quader B:

$$C_A(B) = 36\,\mathrm{Mult} + 24\,\mathrm{Add}\,.$$

Abstandsberechnung

Betrachten wir zunächst die durch den Quader $B_i = \mathrm{OBB}(\mathbf{c}_i, \mathbf{D}_i, \overline{\mathbf{d}}_i)$, $i \in \{1, 2\}$, definierte Punktmenge:

$$\begin{aligned} B_i &= \big\{\mathbf{D}_i\mathbf{x} + \mathbf{c}_i \,\big|\, -\overline{\mathbf{d}}_i \le \mathbf{x} \le \overline{\mathbf{d}}_i\big\} \\ &= \big\{\mathbf{x} \,\big|\, \mathbf{D}_i^T\mathbf{c}_i - \overline{\mathbf{d}}_i \le \mathbf{D}_i^T\mathbf{x} \le \mathbf{D}_i^T\mathbf{c}_i + \overline{\mathbf{d}}_i\big\} \\ &= \big\{\mathbf{x} \,\big|\, \mathbf{A}_i\mathbf{x} \le \mathbf{A}_i\mathbf{c}_i + \overline{\mathbf{a}}_i\big\}, \quad i \in 1, 2\,, \end{aligned}$$

mit $\mathbf{A}_i := \begin{bmatrix} \mathbf{D}_i^T \\ -\mathbf{D}_i^T \end{bmatrix}$ und $\overline{\mathbf{a}}_i := \begin{bmatrix} \overline{\mathbf{d}}_i \\ \overline{\mathbf{d}}_i \end{bmatrix}$.
Der quadratische Euklidische Abstand zweier Quader B_1, B_2 ergibt sich somit unter Berücksichtigung der Aktualisierungsabbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ als

$$\begin{aligned} \delta^2\big(\tau(B_1), B_2\big) &= \min\big\{\|\mathbf{R}\mathbf{x}_1 + \mathbf{t} - \mathbf{x}_2\|^2 \,\big|\, \mathbf{A}_1\mathbf{x}_1 \le \overline{\mathbf{b}}_1,\, \mathbf{A}_2\mathbf{x}_2 \le \overline{\mathbf{b}}_2\big\} \\ &= \min\big\{(\mathbf{R}\mathbf{x}_1 - \mathbf{x}_2)^2 \,\big|\, \mathbf{A}_1\mathbf{x}_1 \le \overline{\mathbf{b}}_1,\, \mathbf{A}_2(\mathbf{x}_2 + \mathbf{t}) \le \overline{\mathbf{b}}_2\big\} \\ &= \min\big\{(\mathbf{R}\mathbf{x}_1 - \mathbf{x}_2)^2 \,\big|\, \mathbf{A}_1\mathbf{x}_1 \le \overline{\mathbf{b}}_1,\, \mathbf{A}_2\mathbf{x}_2 \le \overline{\mathbf{b}}_2 - \mathbf{A}_2\mathbf{t}\big\}\,, \end{aligned}$$

mit $\overline{\mathbf{b}}_i := \mathbf{A}_i\mathbf{c}_i + \overline{\mathbf{a}}_i$.
Wir erhalten somit ein Optimierungsproblem mit quadratischer Zielfunktion und linearen Nebenbedingungen. Algorithmen der nichtlinearen Programmierung arbeiten

jedoch auf kleinen Probleminstanzen, wie sie beispielsweise durch Quader definiert werden, ineffizient, so daß wir uns nach anderen Verfahren umsehen müssen.

Da es sich bei Quadern um konvexe Polyeder handelt, können wir die in 3.2 vorgestellten Verfahren zur Abstandsberechnung einsetzen. Doch auch diese Algorithmen eignen sich primär für konvexe Körper hoher Komplexität, die man beispielsweise durch eine günstige konvexe Zerlegung des Gesamtkörpers erhält. Bedenkt man nun, daß der Branch-and-Bound-Algorithmus den Hüllkörperabstand für jedes betrachtete Paar von Flächenmengen berechnet, so ist der Einsatz solcher „allgemeiner" Verfahren verglichen mit dem Kugeltest des vorangegangenen Abschnittes zu teuer.

Aufgrund der geringen Zahl von Seitenflächen bietet sich zunächst ein *brute-force-Algorithmus* an, der den in 3.4 vorgestellten Abstandstest zweier Polygone zur Berechnung des Quaderabstands verwendet:

$$\delta(B_1, B_2) = \min\left\{\delta(f_1, f_2) \,\middle|\, (f_1, f_2) \in \mathcal{F}(B_1) \times \mathcal{F}(B_2)\right\}.$$

Der Algorithmus von MEYER [Mey86] berücksichtigt die Besonderheiten der Quadergeometrie, um dieses naive Verfahren effizient zu gestalten.

Der Kanten-Klassifikations-Algorithmus von MEYER

Eine deutliche Verbesserung des gerade beschriebenen, naiven Verfahrens stellt der Algorithmus von MEYER [Mey86] dar. Auch hier werden die elementaren Kante-Kante- und Punkt-Fläche-Tests verwendet, um die Distanz der Quader zu bestimmen. Das Verfahren nutzt jedoch die besondere Geometrie des Quaders aus, um die Abstandsberechnung zu beschleunigen.

Betrachtet man die unterstützenden Ebenen der Seitenflächen eines Quaders, so induzieren diese eine Unterteilung des $\mathbb{R}^3$ in die 27 Voronoiregionen, wie Abbildung 3.23 verdeutlicht. Es lassen sich dabei vier Typen von Regionen unterscheiden:

1. *Die Knotenregion* (Abbildung 3.23a)
   Sie geht von jedem der 8 Eckpunkte des Quaders aus und hat genau einen Punkt mit dem Quader gemeinsam, nämlich den jeweiligen Eckpunkt.

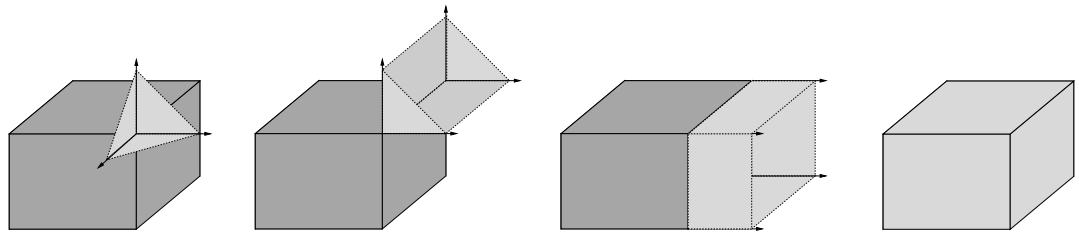
2. *Die Kantenregion* (Abbildung 3.23b)
   Jede der 12 Kanten des Quaders definiert eine Region, die den Quader ausschließlich an dieser Kante schneidet.

3. *Die Flächenregion* (Abbildung 3.23c)
   Von jeder der 6 Seitenflächen geht eine weitere Region aus, die mit dem Quader lediglich diese Punktmenge teilt.

4. *Die innere Region* (Abbildung 3.23c)
   Die letzte Region entspricht dem Quader selbst, bzw. seinem Inneren.

Das Einsetzen eines Eckpunkts von B_1 in die Ebenengleichungen des Quaders B_2 gestattet die Klassifikation dieses Punktes bezüglich seiner Regionenzugehörigkeit. Die Definition der Regionen kann dabei so gestaltet werden, daß die Zugehörigkeitsbeziehung eindeutig ist.

Abbildung 3.23 Die internen und externen Voronoiregionen eines Quaders.



(a) Knotenregion (b) Kantenregion. (c) Flächenregion. (d) Innere Region.

In dem Verfahren von MEYER wird zunächst getestet, ob B_1 den Quader B_2 schneidet. Falls ein Eckpunkt von B_1 in der Region 4 liegt, so müssen sich die beiden Quader durchdringen und wir können den Abstand entsprechend unserer Konvention als Null angeben.
Andernfalls versuchen wir, den minimalen Abstand zwischen den beiden Quadern zu ermitteln, wobei wir eine Durchdringung jedoch nicht ausschließen können.

Satz 3.9. *Der minimale Abstand wird*

(i) zwischen einer Kante von B_1 *und dem Quader* B_2 **oder**

(ii) zwischen einem Eckpunkt von B_2 *und einer Fläche von* B_1

angenommen. Es gilt somit:

$$\delta(B_1, B_2) = \begin{cases} \min\left\{ \min\limits_{e_1 \in \mathcal{E}(B_1)} \delta(e_1, B_2), \min\limits_{v_2 \in \mathcal{V}(B_2)} \delta(\mathcal{F}(B_1), v_2) \right\} & \textit{falls } B_1 \bigcap B_2 \neq \emptyset\,; \\ 0 & \textit{sonst}\,. \end{cases}$$

Beweis. Die Korrektheit dieser Beobachtung ist offensichtlich, wenn wir uns vor Augen führen, daß alle Kante-Kante-Abstände sowie die Abstände der Eckpunkte von B_1 zu den Seitenflächen von B_2 von Fall (i) erfaßt werden und in Fall (ii) die Distanzen der Eckpunkte von B_2 zu den Seitenflächen von B_1 berücksichtigt werden. Somit werden alle Tests durchgeführt, die zur Bestimmung des Abstands zweier Polyeder (vgl. Abschnitt 3.4) erforderlich sind. □

Für Fall (ii) kann der Autor gegenüber unserer allgemeinen Vorgehensweise in 3.4 keine effizienzsteigernden Vorschläge machen. Im ersten Fall erweist sich jedoch die gerade vorgestellte *Raumpartitionierung* als äußerst hilfreich. Für jeden Eckpunkt der Kante wird die Region bestimmt, in der sich der Punkt befindet. Werden die beiden Eckpunkte identisch klassifiziert, so liegt die Kante vollständig im Inneren der entsprechenden Region und der Abstand zwischen Kante und Quader kann regionenspezifisch ermittelt werden.

Falls die Eckpunkte der Kante unterschiedlichen Regionen zugeteilt sind, wird die Kante zerschnitten (*Kantensplitting*), so daß jedes ihrer Teilsegmente in genau einer Region liegt. Fällt dabei ein Segment in das Innere des Quaders, so liegt eine Durchdringung vor und wir erhalten als Abstand der beiden Quader den Wert Null.

Wir wollen uns nun der *regionenspezifischen Abstandsberechnung* zuwenden. Hierzu muß die Distanz der Kante bzw. des Teilsegmentes e_1 zu dem Oberflächenelement von B_2, das die Region definiert, bestimmt werden. Unsere Fallunterscheidung orientiert sich daher am Typ der Region:

1. Die Region zu Knoten v_2: $\mathcal{R}(v_2)$
   In diesem Fall muß der Abstand zwischen einem Geradensegment e_1 und dem Eckpunkt v_2 von B_2 ermittelt werden:

   $$\delta(e_1, B_2) = \delta(e_1, v_2) \, .$$

   Der elementare Punkt-Kante-Test in Abschnitt 3.4.2 hat gezeigt, wie dieser Abstand berechnet werden kann.

2. Die Region zu Kante e_2: $\mathcal{R}(e_2)$
   Den Abstand zweier Geradensegmente können wir ebenfalls aufgrund unserer Betrachtungen in Abschnitt 3.4.1 bestimmen. Es gilt somit:

   $$\delta(e_1, B_2) = \delta(e_1, e_2) \, .$$

3. Die Region zu Fläche f_2: $\mathcal{R}(f_2)$
   Da es sich bei e_1 um ein Geradensegment handelt, wird der minimale Abstand von e_1 zu f_2 an einem der beiden Endpunkte von e_1 angenommen. Die Raumpartitionierung erlaubt den Punkt-Fläche-Test aus Abschnitt 3.4.3 zu beschleunigen. Es ist in dieser Situation völlig ausreichend, die Länge des Lotes der Endpunkte auf f_2 zu bestimmen, wobei der kleinere der beiden Werte den Abstand der Kante zu dem Quader definiert. Der Test, ob der Lotfußpunkt innerhalb der Seitenfläche liegt, entfällt durch die Kenntnis der Regionenzugehörigkeit von e_1. Wir erhalten somit:

   $$\delta(e_1, B_2) = \min \left\{ \delta\big(a_1, \Sigma(f_2)\big), \; \delta\big(a_2, \Sigma(f_2)\big) \right\} ,$$

   wobei $\Sigma(f_2)$ die unterstützende Ebene von f_2 bezeichnet.

Die Korrektheit der regionenspezifischen Abstandsberechnung mittels Kantensplitting, folgt aus der Erkenntnis, daß der minimale Abstand eines Quaders zu einem außerhalb gelegenen Punkt zwischen dem Punkt selbst und dem Oberflächenelement angenommen wird, das die Region definiert, in die der Punkt fällt.

Die Umsetzung des Kantenklassifikationsalgorithmus'

Bei der Umsetzung des Verfahrens müssen drei wesentliche *Effizienzfallen* beachtet werden:

1. Knotenklassifikation durch Einsetzen in die Ebenengleichungen
2. naives Kantensplitting
3. regionenspezifische Abstandsberechnung ohne Berücksichtigung der Quadergeometrie

Ein Schlüssel zur Effizienzsteigerung ist eine geschickte Koordinatentransformation, die uns erlaubt, viele Berechnungen anhand einfacher Koordinatenvergleiche durchzuführen. Dazu wechseln wir das Koordinatensystem derart, daß jeweils einer der beiden Quader im Nullpunkt zentriert liegt und seine Seitenflächen senkrecht auf den Koordinatenachsen stehen. Wir werden zunächst den Quader $B_2(\mathbf{c}_2, \mathbf{D}_2, \overline{\mathbf{d}}_2)$ in das gerade beschriebene Koordinatensystem transformieren. Die gesuchte Abbildung σ ist folgendermaßen definiert:

$$\begin{aligned} \sigma : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{x} &\mapsto \mathbf{D}_2^T(\mathbf{x} - \mathbf{c}_2) = \mathbf{D}_2^T\mathbf{x} - \mathbf{D}_2^T\mathbf{c}_2 \,. \end{aligned}$$

Intuitiv handelt es sich bei σ um die Umkehrtransformation der Abbildung, die den Quader B_2 aus einer nullpunktzentrierten Iso-Box gleicher Ausdehnung $\overline{\mathbf{d}}_2$ entstehen läßt. Es bietet sich nun an, die Aktualisierung τ von B_1 und seine Transformation in das Zielkoordinatensystem zu einer Abbildung $\rho = \sigma \circ \tau$ zusammenzufassen:

$$\begin{aligned} \rho : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{x} &\mapsto \mathbf{D}_2^T\big[\,(\mathbf{R}(\mathbf{D}_1\mathbf{x} + \mathbf{c}_1) + \mathbf{t}) - \mathbf{c}_2\big] \\ &\quad = \mathbf{D}_2^T\mathbf{R}\mathbf{D}_1\mathbf{x} + \mathbf{D}_2^T(\mathbf{R}\mathbf{c}_1 + \mathbf{t} - \mathbf{c}_2) \\ &\quad = \mathbf{P}\mathbf{x} + \mathbf{p} \,, \end{aligned}$$

mit $\mathbf{P} := \mathbf{D}_2^T\mathbf{R}\mathbf{D}_1$ und $\mathbf{p} := \mathbf{D}_2^T(\mathbf{R}\mathbf{c}_1 + \mathbf{t} - \mathbf{c}_2)$.
Die Umkehrabbildung $\rho^{-1} : \mathbf{x} \mapsto \mathbf{P}^T\mathbf{x} - \mathbf{P}^T\mathbf{p}$ transformiert somit B_2 in das nullpunktzentrierte und achsenorientierte Koordinatensystem der Box B_1.

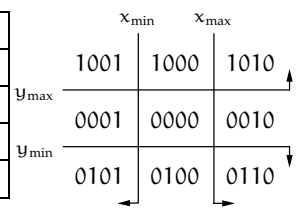
Klassifikation der Eckpunkte und Kanten

Um die Eckpunkte effizient klassifizieren und die Kanten geschickt teilen zu können, verwenden wir einige klassische Ideen aus dem Bereich der Computergraphik. Eine grundlegende Operation in diesem Gebiet ist das Abschneiden von Geradensegmenten bezüglich eines Polygons. Dabei stellt das *Clipping* gegenüber einem rechteckigen Bereich einen wichtigen Spezialfall dar, der beispielsweise zur Bestimmung der in einem Bildfenster sichtbaren Objektteile angewendet wird. Da die analytische Schnittpunktberechnung zwischen Geradensegment und Clipfenster eine in diesem Umfeld teure Operation ist, versucht der Algorithmus von COHEN UND SUTHERLAND [FvDFH96], diese Berechnung möglichst zu vermeiden.

Dazu unterteilt das Verfahren die Bildschirmebene in neun Regionen und ordnet jeder Region eine Bitzahl $\overline{c}$ zu, die die Lage der Region bezüglich des Clipfensters $(x_{min}, x_{max}, y_{min}, y_{max})$ charakterisiert. Die Kodierungsfunktion, die eine effiziente

Abbildung 3.24 Kodierung der zweidimensionalen Raumpartitionierung.

Bit	*gesetzt, falls*
3	$y \geq y_{max}$
2	$y < y_{min}$
1	$x \geq x_{max}$
0	$x < x_{min}$



Lokalisation der Segmentendpunkte (x, y) erlaubt, ist durch die Tabelle 3.24 definiert. Die zugehörige Bereichskodierung ist in der rechten Abbildung veranschaulicht. Durch einfache Bitoperationen kann nun bestimmt werden, ob sich eine Kante vollständig innerhalb bzw. außerhalb des Fensterbereichs befindet und welche Fenstergerade eine Teilung der Kante induziert, so daß mindestens ein Teilsegment vollständig außerhalb des Clipbereichs liegt.

Wir wollen nun die Idee des Algorithmus' von COHEN UND SUTHERLAND auf unser Problem übertragen. Statt des 4-Bit-Binärcodes zur Punktlokalisation in der Ebene, verwenden wir eine 6-Bit-Kodierung zur Charakterisierung der von dem Quader erzeugten 27 Regionen des $\mathbb{R}^3$. Jedes Bit trifft eine Aussage, ob der zu klassifizierende Punkt innerhalb oder außerhalb der mit dem Bit assoziierten Halbgeraden liegt (vgl. Abbildung 3.25). Die Kodierungsfunktion $c : \mathcal{V}(B) \to \{0,1\}^6$ wird durch die Tabelle in Abbildung 3.25 definiert.

Wir wollen nun zeigen, daß man anhand der Kodierung $\overline{c} = c(\boldsymbol{v}_1)$ des Endpunkts v_1 von B_1 sowohl den Regionentyp als auch das Oberflächenelement, das die Region definiert, in der v_1 liegt, eindeutig bestimmen kann.

Beobachtung 2. Ist $\overline{c}$ die 6-Bit-Zahl, die einen Punkt p, entsprechend der in 3.25 definierten Kodierungsfunktion beschreibt, dann ist die Region, in der p bezüglich der quaderinduzierten Raumpartitionierung liegt, eindeutig durch die Zahl der gesetzten Bits $n_B(\overline{c})$ in $\overline{c}$ bestimmt.

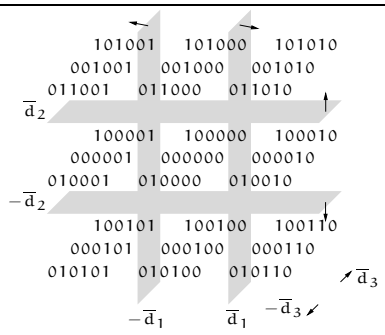
$n_B(\overline{c})$	Regionentyp
0	innere Region (Typ 4)
1	Flächenregion (Typ 3)
2	Kantenregion (Typ 2)
3	Knotenregion (Typ 1)

Der Grund hierfür ist, daß ein Bit in $\overline{c}$ genau dann gesetzt wird, wenn der Punkt p außerhalb der durch das Bit beschriebenen Halbebene liegt. Wir können somit eine Funktion ζ definieren, die der Kodierung eines Punktes die Region zuweist, der er angehört.

$$\begin{aligned} \zeta : \{0,1\}^6 &\to \{1,2,3,4\} \\ \overline{c} &\mapsto 4 - n_B(\overline{c}) \,. \end{aligned}$$

Abbildung 3.25 Kodierung der dreidimensionalen Raumpartitionierung.

Bit	*gesetzt, falls*
5	$x_3 \geq \overline{d}_3$
4	$x_3 < -\overline{d}_3$
3	$x_2 \geq \overline{d}_2$
2	$x_2 < -\overline{d}_2$
1	$x_1 \geq \overline{d}_1$
0	$x_1 < -\overline{d}_1$



Im folgenden werden wir Bitoperationen auf der Binärkodierung $\overline{c}$ des Punktes p durchführen. Die Symbole $\otimes$ und $\oplus$ bezeichnen die Binäroperatoren AND bzw. XOR, die auf Bitvektoren komponentenweise arbeiten. Den Shiftoperator, der von rechts bzw. links eine Null in den Bitvektor schiebt, wollen wir durch das Symbol $\ll$ bzw. $\gg$ darstellen. Die Shiftoperation kann i-mal hintereinander auf $\overline{c}$ angewendet werden, was wir mit der Schreibweise $\overline{c} \ll i$ bzw. $i \gg \overline{c}$ bezeichnen wollen. Zusätzlich wollen wir die folgenden Bitvektoren als Masken verwenden: $m_0 := (0,0,0,0,0,0)$, $m_1 := (0,0,0,0,0,1)$ und $m_3 := (0,0,0,0,1,1)$.

Beobachtung 3. Liegt ein Punkt p in einer der 6 Flächenregionen des Quader B, so kann die entsprechende Fläche von B anhand der Kodierung $\overline{c}$ identifiziert werden:

$$\begin{aligned} \xi_3 : \{0,1\}^6 &\rightarrow \mathcal{F}(B) \\ \overline{c} &\mapsto f_n \,, \end{aligned}$$

wobei f_n die Fläche von B ist, die durch den Normalenvektor $\mathbf{n} = (n_1, n_2, n_3)^T$ definiert ist:

$$n_i = \begin{cases} -1 & \text{falls } \overline{c} \otimes m_1 \ll 2(i-1) \neq m_0 \,; \\ 1 & \text{falls } \overline{c} \otimes m_1 \ll 2(i-1)+1 \neq m_0 \quad 1 \leq i \leq 3 \,. \\ 0 & \text{sonst} \,. \end{cases}$$

Diese Darstellung ist korrekt, da in $\overline{c}$ aufgrund der Regionenlage von p lediglich ein Bit gesetzt ist. Diese Eins wurde erzeugt, weil sich p außerhalb der Halbebene befindet, auf deren Begrenzung f_n liegt. Somit bestimmt die Position des gesetzten Bits die gesuchte Fläche.

Beobachtung 4. Befindet sich der Punkt p in einer der 8 Knotenregionen des Quaders B, so kann der entsprechende Endpunkt q anhand der Kodierung $\overline{c}$ von p eindeutig bestimmt werden.

$$\begin{aligned} \xi_1 : \{0,1\}^6 &\rightarrow \mathcal{V}(B) \\ \overline{c} &\mapsto q \,, \end{aligned}$$

wobei sich der Ortsvektor $\mathbf{q} = (q_1, q_2, q_3)^T$ von q folgendermaßen ergibt:

$$q_i = \begin{cases} -\overline{d}_i & \text{falls } \overline{c} \otimes m_1 \ll 2(i-1) \neq m_0\,; \\ \overline{d}_i & \text{sonst}\,. \end{cases} \qquad 1 \leq i \leq 3\,.$$

Dies gilt, da die drei Halbebenen, außerhalb derer p liegt, die Koordinaten des die Knotenregion definierenden Eckpunktes q bestimmen.

Beobachtung 5. Liegt der Punkt p in einer der 12 Kantenregionen des Quaders B, so kann die entsprechende Kante e anhand der Kodierung $\overline{c}$ identifiziert werden. Die Eckpunkte a und b von e können folgendermaßen ermittelt werden:

$$\begin{aligned} \xi_2 : \{0,1\}^6 &\rightarrow \mathcal{E}(B) \\ \overline{c} &\mapsto e = (a, b), \\ \text{mit} \quad a &= \xi_1(\overline{c}_1) \\ b &= \xi_1(\overline{c}_2) \\ \text{und} \quad \overline{c}_1 &:= \overline{c} \oplus m_1 \ll 2(j-1) \\ \overline{c}_2 &:= \overline{c} \oplus m_1 \ll 2(j-1)+1\,, \end{aligned}$$

wobei j der eindeutige Index aus $\{1,2,3\}$ ist, für den gilt: $\overline{c} \otimes m_3 \ll 2(j-1) = m_0$.
Der Richtungsvektor $\boldsymbol{\nu}$ der Kante e entspricht einem Einheitsvektor des $\mathbb{R}^3$:

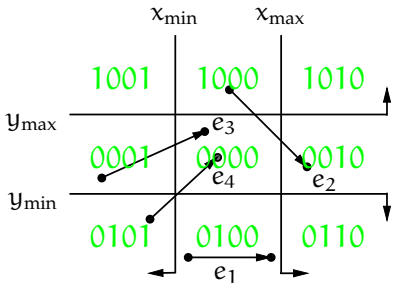
$$\nu_i = \begin{cases} 1 & \text{falls } \overline{c} \otimes m_3 \ll 2(i-1) = m_0\,; \\ 0 & \text{sonst}\,. \end{cases} \qquad 1 \leq i \leq 3\,.$$

Die Beobachtung folgt aus der Einsicht, daß sich die Endpunkte einer Kante lediglich in der Lage bezüglich zweier Halbebenen unterscheiden. Bei den Begrenzungsebenen dieser Halbräume handelt es sich um die unterstützenden Ebenen zweier gegenüberliegender Flächen. Die Kante e liegt innerhalb dieser beiden Halbräume. Für alle anderen Paare von parallelen Halbräumen gilt, daß e sich innerhalb des einen und außerhalb des anderen befindet. Somit sehen wir:

$$\overline{c} = \begin{cases} (*,*,*,*,0,0) & \text{falls } e \parallel \mathbf{d}_1 = \boldsymbol{e}_1\,; \\ (*,*,0,0,*,*) & \text{falls } e \parallel \mathbf{d}_2 = \boldsymbol{e}_2\,; \\ (0,0,*,*,*,*) & \text{falls } e \parallel \mathbf{d}_3 = \boldsymbol{e}_3\,, \end{cases}$$

wobei $\boldsymbol{e}_i$ den i-ten Einheitsvektor, $1 \leq i \leq 3$, bezeichnet.
Die Kantenregion können wir also bestimmen, indem wir die Maske m_3 verwenden und das Paar der gesetzten Bits jeweils zwei Stellen verschieben. Der Ausdruck $\overline{c} \otimes m_3 \ll 2(j-1)$ ist genau dann gleich m_0, wenn Bit $2(j-1)$ und $2(j-1)+1$ nicht gesetzt sind. Setzen wir nun das höherwertigere der beiden aufeinanderfolgenden ungesetzten Bits, so erhalten wir die Bereichskodierung des Kantenendpunktes b, andernfalls des Anfangspunktes a. Mit Hilfe des vorgestellten Klassifikationsschemas für Raumpunkte lassen sich auch Kanten bezüglich ihrer Regionenzugehörigkeit charakterisieren. Kanten können selbstverständlich mehrere Regionen durchlaufen, so daß ihre Unterteilung

Abbildung 3.26 Ein zweidimensionales Beispiel zur Kantenunterteilung.



zur Verwendung der regionenspezifischen Abstandsberechnung unumgänglich ist.

Unterteilung von Kanten

Die Klassifikation von Endpunkten bezüglich Regionen erweist sich auch bei der Unterteilung der Kanten als äußerst hilfreich. Wir wollen die grundlegenden Ideen zunächst am zweidimensionalen Fall, der in Abbildung 3.26 dargestellt ist, erläutern. Sei $e = (a, b)$ die zu betrachtende Kante, dann können wir die beiden folgenden Fälle unterscheiden.

1. Fall: $c(\mathbf{a}) = c(\mathbf{b})$
   Falls beide Endpunkte der gleichen Region zugeordnet werden, dann liegt die Kante $e = (a, b)$ vollständig in der Region $\mathcal{R}(a) = \mathcal{R}(b)$. Eine Unterteilung der Kante ist somit nicht erforderlich. Die Kante e_1 in Abbildung 3.26 würde auf diese Weise klassifiziert werden.

2. Fall: $c(\mathbf{a}) \neq c(\mathbf{b})$
   Liegen die Eckpunkte der Kanten in unterschiedlichen Regionen, so ist eine Unterteilung der Kante erforderlich. Wir wollen die Kante e dabei nicht wirklich in Segmente zerlegen, sondern lediglich bestimmen, welche Regionen von ihr durchlaufen werden. Anschließend ermitteln wir den Abstand zwischen den Oberflächenelementen, die die entsprechenden Regionen definieren und der gesamten Kante. Betrachtet man bei der regionenspezifischen Abstandsberechnung die Kante statt dem die Region durchlaufenden Teilsegment, so läuft man lediglich im Fall einer Flächenregion Gefahr, einen falschen Kante-Quader-Abstand zu bestimmen. Hier sind wir entgegen unseren bisherigen Aussagen gezwungen, zu überprüfen, ob die Projektion des abstandsminimierenden Endpunkts der Kante innerhalb der Seitenfläche des Quaders liegt. Der Vorteil dieser Vorgehensweise ist jedoch, daß wir die Verfahren aus 3.4 einsetzen können, nachdem wir sie im nächsten Abschnitt auf die Quadergeometrie optimiert haben.

Im Rahmen der Kantenunterteilung ermitteln wir zunächst die Halbebenen, die von der Kante durchstoßen werden. Wir erinnern uns daran, daß jedes Bit der Endpunktkodierung mit einer Halbebene assoziiert werden kann und daß der Wert dieser Bits

angibt, ob der Punkt innerhalb bzw. außerhalb des Halbraums liegt. Somit bedeutet ein Bitwechsel zwischen Anfangs- und Endpunkt, daß die begrenzende Ebene von der Kante durchstoßen wird. Wir berechnen daher den Bitvektor, der alle Bitwechsel zwischen Anfangs- und Endpunktkodierung mit einem gesetzten Bit an der entsprechenden Position protokolliert. Diesen erhalten wir als:

$$s := c(\mathbf{a}) \oplus c(\mathbf{b}) \,.$$

Im Fall von Kante e_2 in Abbildung 3.26 ergibt sich $s = (1,0,1,0)$. Offensichtlich durchstößt die Kante die Halbebene des ersten und dritten Bits, nämlich $\mathcal{H}_1 = \{\mathbf{x} \in \mathbb{R}^3 \,|\, x_1 \geq \overline{d}_1\}$ und $\mathcal{H}_3 = \{\mathbf{x} \in \mathbb{R}^3 \,|\, x_2 \geq \overline{d}_2\}$.

Was wir an dieser Darstellung jedoch nicht ersehen können, ist die Reihenfolge, in der die Regionen durchlaufen werden, d.h. die begrenzenden Ebenen überschritten werden. Falls wir genau einen Bitwechsel zwischen Anfangs- und Endpunktkodierung haben, so ist eine Berechnung der Durchstoßungspunkte nicht erforderlich, da wir alle von der Kante durchlaufenen Regionen bereits kennen. Andernfalls sind wir jedoch gezwungen, die Parameterwerte der Schnittpunkte mit den begrenzenden Ebenen zu ermitteln und nichtfallend zu sortieren. Da alle sechs Ebenen Σ_i, $i = 0,\ldots,5$ senkrecht auf einer Koordinatenachse stehen, ergibt sich beispielsweise für die Begrenzungsebene Σ_0 des Halbraums $\mathcal{H}_0 = \{\mathbf{x} \in \mathbb{R}^3 \,|\, x_1 \geq -\overline{d}_1\}$:

$$\mu_0 = \frac{-\overline{d}_1 - a_1}{v_1}$$

bzw. allgemein für die Halbebene $\mathcal{H}_i$, $i \in \{0,\ldots,5\}$:

$$\mu_i = \frac{\tilde{d}_j - a_j}{v_j} \,,$$

$$\text{mit } j := \left\lfloor \frac{i}{2} \right\rfloor + 1 \qquad \text{und} \qquad \tilde{d}_j := \begin{cases} -\overline{d}_j & \text{falls } i \bmod 2 = 0\,; \\ \overline{d}_j & \text{sonst}\,. \end{cases}$$

In unserem zweidimensionalen Beispiel besitzt die Kante e_2 zwei Durchstoßungspunkte mit den Begrenzungsebenen Σ_1 und Σ_3, die durch die Parameterwerte $\mu_3 < \mu_1$ charakterisiert werden. Kippen wir die entsprechenden Bits des s-Vektors in der Reihenfolge der sortierten Parameterwerte, so erhalten wir sukzessive die Kodierungen der Regionen, die von der Kante durchlaufen werden. Im Fall von e_2 ergeben sich durch Wechsel des dritten und anschließend des ersten Bits die folgenden Regionen:

$$c(\mathbf{a}) = (1,0,0,0) \xrightarrow{3.\text{Bit}} (0,0,0,0) \xrightarrow{1.\text{Bit}} (0,0,1,0) = c(\mathbf{b}) \,.$$

Die Aufzählung der durchlaufenen Regionen im Rahmen des Kantensplittings erlaubt es zudem, eine Durchdringung der Quader zu erkennen. Eine solche liegt vor, falls sich ein Endpunkt innerhalb der inneren Region befindet oder eine Kante diese Region durchläuft. Die Kanten e_3 und e_2 in Abbildung 3.26 veranschaulichen die beiden Fälle.

Durchstößt eine Kante zwei oder gar drei Halbebenen (Kante e_4 in Abbildung 3.26), so liegt eine entartete Situation vor. Sie kann dadurch identifiziert werden, daß die Parameterwerte der Schnittpunkte μ_i, μ_j und eventuell μ_k identisch sind. Im Fall von e_4 weist unser Verfahren die Kante den drei Regionen $(0,1,0,1)$, $(0,1,0,0)$ und $(0,0,0,0)$ zu, obwohl dies für die zweite nicht notwendig gewesen wäre. Eine effiziente Implementierung sollte eine solche Redundanz jedoch vermeiden.

Es stellt sich abschließend die Frage, wie viele Kantenunterteilungen maximal durchgeführt werden müssen.

Satz 3.10. *Eine Kante durchläuft maximal sieben Regionen, d.h. es müssen maximal sechs Kantenunterteilungen durchgeführt werden. Diese obere Schranke ist scharf.*

Beweis. Jeder Regionenwechsel bedeutet einen zusätzlichen Bitwechsel in der Kodierung des Startpunktes. Hat ein Bit seinen Wert geändert, so kann dieser Wechsel bei der weiteren Verfolgung des Kantenverlaufs nicht mehr rückgängig gemacht werden, da die Kante eine Ebene nur einmal durchstoßen kann. Somit kann es maximal sechs Bitwechsel geben, die wie oben beschrieben, mit einem Regionenwechsel und somit einer Kantenunterteilung einhergehen.

Das folgende Beispiel zeigt, daß diese Schranke scharf ist. Sei $\overline{d} = (1,1,1)^T$ die Ausdehnung des Quaders und $e(\mu) = \mathbf{a} + \mu\boldsymbol{\nu}$, $\mu \in [0,1]$ die Parameterdarstellung der Kante, wobei $\mathbf{a} = (-2,-2,-2)^T$ und $\mathbf{b} = (8,5,2)^T$ gilt. Mit $\boldsymbol{\nu} = (10,7,4)$ erhalten wir als Parameterwerte der Durchstoßungspunkte:

$$\mu_0 = \frac{1}{10} < \mu_2 = \frac{1}{7} < \mu_4 = \frac{1}{4} < \mu_1 = \frac{3}{10} < \mu_3 = \frac{3}{7} < \mu_5 = \frac{3}{4}\,.$$

□

Der folgende Satz faßt die Abstandsberechnung unter Berücksichtigung der Kantenunterteilung zusammen.

Satz 3.11. *Sei* $s := c(\mathbf{a}) \oplus c(\mathbf{b})$ *der Bitvektor, der die Durchstoßungspunkte der Kante* $e = (a,b)$ *mit den Begrenzungsebenen* $\Sigma_{i_1},\ldots,\Sigma_{i_k}$, $1 \le k \le 6$ *identifiziert, d.h.*

$$s = (s_0,\ldots,s_5), \qquad \textit{mit} \quad s_i = \begin{cases} 1 & \textit{falls } e \cap \Sigma_i \neq \emptyset\,; \\ 0 & \textit{sonst}\,, \end{cases} \qquad 0 \le i < 6\,.$$

Ferner sei $\sigma_k = (j_1,\ldots,j_k)$ *die aufsteigend sortierende Permutation der Schnittpunktparameter* $\mu_{i_1},\ldots,\mu_{i_k}$. *Dann lautet die Folge* b_{k+1} *der von der Kante* e *durchlaufenen Regionen, die entsprechend unseres Klassifikationsschemas kodiert sind:*

$$b_{k+1}(e) = \big(c_1 = c(\mathbf{a}), c_2 = c_1 \oplus m_{j_1},\ldots,c_{k+1} = c_k \oplus m_{j_k} = c(\mathbf{b})\big)\,,$$

wobei m_{j_l} *der 6-Bit-Vektor ist, bei dem ausschließlich das* j_l*-te Bit gesetzt ist,* $0 \le l < 6$.
Der Abstand von e *zu dem achsenorientierten Quader* B *ergibt sich dann als:*

$$\delta(e,B) = \min_{c \in \{c_1,\ldots c_{k+1}\}} \delta_c(e)\,,$$

wobei $\delta_c(e)$ *den Euklidischen Abstand zwischen* e *und dem Oberflächenelement* $\xi_{\zeta(c)}(c)$ *unter der Annahme, daß* e *in dessen Region liegt, bezeichnet.*

Beweis. Die Korrektheit des Satzes ergibt sich unmittelbar aus den Überlegungen dieses Abschnitts. □

Das Verfahren zur Unterteilung einer Kante wird durch Algorithmus 14 beschrieben.

Algorithmus 14 Ein Algorithmus zur regionalen Unterteilung einer Kante.

Eingabe: Eine Kante $e = (a, b)$, gegeben durch ihre kodierten Endpunkte $c_a = c(\mathbf{a})$ $c_b = c(\mathbf{b})$ und ein nullpunktzentrierter, achsenorientierter Quader $B = (\mathbf{c}, \mathbf{D}, \overline{\mathbf{d}})$.
Ausgabe: Die Folge der durchlaufenen Regionen $b_{k+1} = (c_1, \ldots, c_{k+1})$, $k \geq 0$.

EDGESPLITTING(c_a, c_b, B)
(1) $s \leftarrow c_a \oplus c_b$
(2) Bestimme $i_1, \ldots, i_k$, $0 \leq k \leq 6$, mit $s_{i_j} = 1$, $1 \leq j \leq k$
(3) **switch** k
(4) **case** 0
(5) **return** (c_a)
(6) **case** 1
(7) **return** (c_a, c_b)
(8) **default**
(9) **for** $l \leftarrow 1$ **to** k
(10) $j \leftarrow (1 \gg i_l) + 1 \qquad (* \quad j = \lfloor \frac{i_l}{2} \rfloor + 1 \quad *)$
(11) **if** $i \bmod 2 = 0$ **then** $\tilde{d}_j \leftarrow -\overline{d}_j$
(12) **else** $\tilde{d}_j \leftarrow \quad \overline{d}_j$
(13) $\mu_{i_l} \leftarrow \frac{\tilde{d}_j - a_j}{v_j}$
(14) $(\mu_{j_1}, \ldots, \mu_{j_k}) \leftarrow$ SORTININCREASINGORDER$((\mu_{i_1}, \ldots, \mu_{i_k}))$
(15) $m \leftarrow (0, \ldots, 0, 1)$
(16) $c_1 \leftarrow c_a$
(17) **for** $l \leftarrow 1$ **to** k
(18) $c_{l+1} \leftarrow c_l \oplus (m \ll j_l)$
(19) **return** $(c_1, \ldots, c_{k+1})$

Die quaderspezifische Abstandsberechnung

Die Effizienz des Verfahrens von MEYER beruht auf der Klassifikation der Quaderkanten bezüglich ihrer Lage zu der anderen Box. Die Zuteilung der Kanten zu den oben

beschriebenen Regionen verhindert, daß Abstände zwischen Oberflächenelementen berechnet werden, die als Lieferanten des Abstandsminimums nicht in Frage kommen. Zudem erlaubt diese Form der Kantenklassifikation, einen Teil der teuren Kante-Kante-Tests im naiven Algorithmus durch günstigere Punkt-Kante-Tests zu ersetzen.

Die Optimierung der Abstandstests bezüglich der Quadergeometrie bietet eine weitere Möglichkeit zur Effizienzsteigerung. Dabei können wir ausnutzen daß stets ein Quader im Nullpunkt zentriert liegt und die Koordinatenachsen auf seinen Seitenflächen senkrecht stehen. In der ersten Phase des Algorithmus', die Fall (i) entspricht, wird B_2 in diese ausgezeichnete Lage transformiert. Bei der Behandlung von Fall (ii) in der sich anschließenden Phase wollen wir davon ausgehen, daß B_1 dieser achsenorientierte, nullpunktzentrierte Quader ist.

Wir betrachten zunächst Fall (i) (regionenspezifische Abstandsberechnung). Der in Abschnitt 3.4.2 vorgestellte Punkt-Kante-Test kann durch die spezielle Lage des Quaders B_2 nicht beschleunigt werden. Für den Fall des Kante-Kante-Tests aus Abschnitt 3.4.1 vereinfachen sich jedoch einzelne Berechnungen, da der Richtungsvektor der betrachteten Kante von B_2, $\boldsymbol{v}_2$, parallel zu einer der Koordinatenachsen ist. Dies wirkt sich insbesondere bei der Bestimmung der Lotfußpunkte 3.7, 3.8 und der nahesten Punkte zwischen den mit den Kanten assoziierten Geraden aus (3.4, 3.5). Deutlicher profitiert der Kante-Fläche-Test, von der achsenorientierten Ausrichtung des Quaders B_2, wie wir im folgenden aufzeigen werden. Die Bezeichnung Kante-Fläche-Test ist nicht ganz präzise, da wir für den Fall, daß die Kante e_1 in die Region der Seitenfläche f_2 von B_2 fällt, nur dann die Distanz zwischen e_1 und f_2 berechnen wollen, wenn der Endpunkt von e_1, der den Abstand zu der unterstützenden Ebene von f_2 minimiert, innerhalb von f_2 liegt. Befindet sich dieser Punkt außerhalb der Fläche, so verläßt e_1 die Flächenregion über eine Kantenregion, in der der minimale Abstand zwischen e_1 und B_2 angenommen wird. Es handelt sich somit um einen Punkt-Fläche-Test, für den zunächst ein Eckpunkt von e_1 ausgewählt werden muß. Da der Normalenvektor $\boldsymbol{n}_2$ von f_2 parallel zu einer Koordinatenachse mit Index i ist, erhalten wir als regionenspezifischen Abstand zwischen e_1 und f_2:

$$\delta_3^2(e_1, f_2) = \begin{cases} (\overline{d}_{2i} + a_{1i})^2 & \text{falls } (v_{1i} \leq 0 \wedge n_{2i} < 0) \wedge \pi_{f_2}(a_1) \in f_2\,; \\ (b_{1i} - \overline{d}_{2i})^2 & \text{falls } (v_{1i} \leq 0 \wedge n_{2i} > 0) \wedge \pi_{f_2}(b_1) \in f_2\,; \\ (\overline{d}_{2i} + b_{1i})^2 & \text{falls } (v_{1i} > 0 \wedge n_{2i} < 0) \wedge \pi_{f_2}(b_1) \in f_2\,; \\ (a_{1i} - \overline{d}_{2i})^2 & \text{falls } (v_{1i} > 0 \wedge n_{2i} > 0) \wedge \pi_{f_2}(a_1) \in f_2\,; \\ \min\limits_{e_2 \in \mathcal{E}(f_2)} \delta_2(e_1, e_2)\,, \end{cases}$$

wobei das Subskript c der Abstandsfunktion darauf hinweist, daß der Distanzwert $\delta_c(e_1, F_2)$ nur dann mit dem Euklidischen Abstand übereinstimmt, falls e_1 in der Region $c = \zeta(F_2)$ des Oberflächenelements F_2 von B_2 liegt.
Der Test, ob die Projektion des abstandsminimierenden Endpunkts von e_1 auf die unterstützende Ebene von f_2 innerhalb der Fläche liegt, kann auf einfache Weise durch Koordinatenvergleiche durchgeführt werden.

Sei $j, k \in \{1,2,3\} \setminus \{i\}$, $j < k$, die Indizes der Koordinatenachsen senkrecht auf $\mathbf{n}_2$ und sei p die Projektion π_{f_2} des abstandsminimierenden Eckpunkts von e_1, dann gilt:

$$p \in f_2 \iff (-\overline{d}_{2j} \leq p_j \leq \overline{d}_{2j} \wedge -\overline{d}_{2k} \leq p_k \leq \overline{d}_{2k}) .$$

Somit erhalten wir als regionenspezifische Abstandsberechnung:

$$\delta_c(e_1) = \begin{cases} \delta(e_1, a_2) & \zeta(c) = 1 \wedge \xi_1(c) = a_2 ; \\ \delta_2(e_1, e_2) & \zeta(c) = 2 \wedge \xi_2(c) = e_2 ; \\ \delta_3(e_1, f_2) & \zeta(c) = 3 \wedge \xi_3(c) = f_2 ; \\ 0 & \zeta(c) = 4 . \end{cases}$$

Abschließend wollen wir uns mit der Optimierung der Abstandstests in Fall (ii) beschäftigen. In dieser zweiten Phase des Algorithmus wird die minimale Distanz der Eckpunkte von B_2 zu den Seitenflächen von B_1 ermittelt. Dazu ist maximal ein Punkt-Fläche-Abstandstest notwendig, wie die folgende Überlegung zeigt. Offensichtlich muß der Abstand eines Eckpunkts a_2 von B_2 zu einer Fläche f_1 von B_1 nur dann berechnet werden, wenn a_2 in der Region von f_1 liegt. Andernfalls befindet sich der naheste Punkt von f_1 zu a_2 auf einer Kante von f_1, was bereits von Fall (i) im Rahmen eines Punkt-Kante-Tests berücksichtigt worden ist. Bezeichnen wir die sechs Seitenflächen des achsenorientierten, nullpunktzentrierten Quaders B_1 über die Richtungen ihrer Normalen mit $f_1^{\pm x}, f_1^{\pm y}$ und $f_1^{\pm z}$, so ergibt sich der quadratische Euklidische Abstand eines Eckpunkts a_2 von B_2 zu den Seitenflächen von B_1 als:

$$\delta^2(\mathcal{F}(B_1), a_2) = \begin{cases} (\overline{d}_{11} + a_{21})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{-x}) ; \\ (a_{21} - \overline{d}_{11})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{x}) ; \\ (\overline{d}_{12} + a_{22})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{-y}) ; \\ (a_{22} - \overline{d}_{12})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{y}) ; \\ (\overline{d}_{13} + a_{23})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{-z}) ; \\ (a_{23} - \overline{d}_{13})^2 & \text{falls } a_2 \in \mathcal{R}(f_1^{z}) ; \\ \min\limits_{e_1 \in \mathcal{E}(B_1)} \delta^2(e_1, a_2) . \end{cases}$$

Algorithmus Wir erhalten somit das durch Algorithmus 15 beschriebene Kantenklassifikationsverfahren zur Abstandsberechnung zweier Quader B_1 und B_2. Das Superskript (i) bezeichnet die Darstellung des Quaders in einem Koordinatensystem, in dem der Quader B_i nullpunktzentriert und achsenorientiert liegt.

3.7.5 Der Abstand zweier Fixed Directions Hulls

Aktualisierung

Im Gegensatz zu den bisher betrachteten Hüllkörpern sind Fixed Directions Hulls in ihrer Hüllkörpereigenschaft ausschließlich unter translatorischen Bewegungen der umhüllten Flächenmenge abgeschlossen. Sobald das eingeschlossene Objekt eine Rotation

Algorithmus 15 Ein Algorithmus zur Abstandsberechnung zweier Quader.

Eingabe: Zwei Quader $B_i(\mathbf{c}_i, \mathbf{D}_i, \overline{\mathbf{d}}_i)$, $i = 1, 2$.
Ausgabe: Der quadratische Euklidische Abstand der beiden Quader.

```
BOXBOXDISTANCE(B_1, B_2)
(1)   d* ← ∞
(2)   B_1^(2) ← P B_1 + p            (* B_1^2 = ρ(B_1) *)
(3)   B_2^(2) ← D_2^T (B_2 − c_2)    (* B_2^2 = σ_2^{-1}(B_2) *)
(4)   foreach e_1 = (a_1, b_1) ∈ E(B_1)
(5)     c_a ← c(a_1)
(6)     c_b ← c(b_1)
(7)     (c_1, ..., c_{k+1}) ← EDGESPLITTING(c_a, c_b)
(8)     for i ← 1 to k + 1
(9)       switch ζ(i)
(10)      case 1
(11)        d ← δ(e_1, ξ_1(c_i))
(12)      case 2
(13)        d ← δ_2(e_1, ξ_2(c_i))
(14)      case 3
(15)        d ← δ_3(e_1, ξ_3(c_i))
(16)      default
(17)        return 0
(18)      d* ← min{d, d*}
(19)  B_2^(1) ← P^T (B_2 − p)        (* B_2^1 = ρ^{-1}(B_2) *)
(20)  B_1^(1) ← D_1^T (B_1 − c_1)    (* B_1^1 = σ_1^{-1}(B_1) *)
(21)  foreach a_2 ∈ V(B_2)
(22)    c_a ← c(a_2)
(23)    if ζ(c_a) = 3
(24)      d ← δ(ξ_3(c_a), a_2)
(25)      d* ← min{d, d*}
(26)  return d*
```

ausführt, ist man gezwungen, die FDH_n bezüglich ihrer Richtungsvektoren neu zu berechnen. Wendet man stattdessen in Analogie zu den bisher betrachteten Primitiven die Rotationsbewegung auf die Hüllkörper der Hierarchie an, so erhält man eine zulässige Approximation des Körpers durch Fixed Directions Hulls, doch die Richtungsvektoren, die die FDH_n definieren, haben sich verändert. Im Rahmen der Abstandsbestimmung werden wir sehen, daß die Effizienz der Kollisions- und Abstandstests zwischen FDHs jedoch gerade auf der Eigenschaft beruht, daß alle Hüllkörper bezüglich der gleichen Menge von Richtungsvektoren ausgedrückt sind. Wir wollen im folgenden verschiedene Verfahren vorstellen, die eine Aktualisierung der FDH_n-Hierarchie durchführen und dabei sicherstellen, daß jeder Hüllkörper bezüglich der gleichen Menge von Rich-

tungsvektoren D dargestellt ist.

1. Das Brute-Force-Verfahren

Da nur die Eckpunkte der *konvexen Hülle* von $\mathcal{F}$, $CH(\mathcal{F})$ als Extrempunkte in den Richtungen aus D in Frage kommen, wird bei dieser Vorgehensweise in der Vorberechnungsphase für jeden Hüllkörper die konvexe Hülle der einzuschließenden Flächenmenge berechnet und deren Eckpunkte in den entsprechenden Knoten der Hierarchie gespeichert. Die Aktualisierung des Hüllkörpers nach einem Bewegungsschritt besteht nun in der Transformation dieser Punktmenge und der anschließenden Bestimmung der Extrempunkte in den vorgegebenen Achsenrichtungen D:

$$FDH_n\big(\tau(\mathcal{F})\big) = FDH_n\big(\tau\big(CH(\mathcal{F})\big)\big) .$$

Dabei nutzt man aus, daß die Eckpunktmenge der konvexen Hülle im allgemeinen sehr viel kleiner ist als die der zugrundeliegenden Flächenmenge. Nachteilig wirkt sich bei dieser Vorgehensweise dagegen aus, daß alle Eckpunkte der konvexen Hülle gespeichert und nach jedem Zeitschritt aktualisiert werden müssen. Da es sich bei dieser Punktmenge im schlimmsten Fall um die vollständige Eckpunktmenge von $\mathcal{F}$ handelt, ist das Verfahren sowohl *speicherplatz-* als auch *rechenzeitaufwendig*.

2. Der Hill-Climbing-Algorithmus

Der Hill-Climbing-Algorithmus setzt in allen Knoten der Hierarchie die *Randrepräsentation* (*b-rep*) der konvexen Hülle voraus. Dieser im Vergleich zum Brute-Force-Verfahren zusätzliche Speicheraufwand erlaubt die aktuellen Extrempunkte bezüglich der vorgegebenen Richtungen durch *lokale Tests* aus den Extrempunkten des letzten Zeitschritts zu bestimmen (vgl. GJK-Variante von CAMERON in Abschnitt 3.2.2).

Der Algorithmus überprüft sukzessive, ob die Extrempunkte in den einzelnen Richtungen aus D nach Durchführung des Bewegungsschritts ihre Extremalitätseigenschaften behalten haben. Dazu werden für jeden Extrempunkt p_i^*, $i = 1, \ldots, n$ die adjazenten Knoten aktualisiert und deren Lage bezüglich der betrachteten Richtung $\mathbf{d}_i$ mit der von p_i^* verglichen. Besitzt einer der Nachbarknoten einen größeren Abstand in Richtung von $\mathbf{d}_i$ zum Ursprung, so ist p_i^* nicht mehr optimal und wir wählen den adjazenten Knoten, für den dieser Abstand maximal ist. Dieser sogenannte *Hill-Climbing-Schritt* wird solange wiederholt, bis der neue Extrempunkt in Richtung von $\mathbf{d}_i$ gefunden ist. Das Verfahren wird durch Algorithmus 16 verdeutlicht.

Das Verfahren ist ein Beispiel für die Ausnutzung *temporärer Kohärenz*. Falls der Zeitschritt zwischen zwei aufeinanderfolgenden Bewegungszuständen klein ist, so ist die im Rahmen der Bewegung durchgeführte Rotation ebenfalls gering und wir finden die neuen Extrempunkte mit wenigen Hill-Climbing-Schritten. Ist m die Anzahl der Eckpunkte der konvexen Hülle, so benötigt das Verfahren im worst-case zwar immer noch $O(m)$ Punktaktualisierungen, doch in der Praxis zeigt sich, daß die Zahl der durchgeführten Hill-Climbing-Schritte für jede Richtung nahezu konstant ist.

Das Approximationsverfahren

Im Gegensatz zu den bisher betrachteten Verfahren berechnet diese Methode nicht die

Algorithmus 16 Der Hill-Climbing-Algorithmus zur Aktualisierung einer FDH_n.

Eingabe: Die zu aktualisierende Fixed Directions Hull $F = \mathrm{FDH}_n(\mathbf{D}, \overline{\mathbf{d}})$ mit den Extrempunkten des letzten Bewegungszustands $(p_1^*, \ldots, p_n^*)$, die konvexe Hülle der zugrundeliegenden Flächenmenge $\mathrm{CH}(\mathcal{F})$ und die Bewegungsabbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$.
Ausgabe: Die aktualisierte Fixed Directions Hull $F' = \mathrm{FDH}_n(\mathbf{D}, \overline{\mathbf{d}}')$.

UPDATEFDH($F, \mathrm{CH}(\mathcal{F}), \mathbf{R}, \mathbf{t}$)
(1) **for** $i \leftarrow 1$ **to** n
(2) $\quad \mathbf{p}^* \leftarrow \mathbf{R}\mathbf{p}_i^* + \mathbf{t}$
(3) $\quad d^* \leftarrow \mathbf{d}_i^T \mathbf{p}_i^*$
(4) $\quad$ **repeat**
(5) $\quad\quad p_i^* \leftarrow p^*$
(6) $\quad\quad$ **foreach** $q \in \mathcal{V}_{\mathrm{adj}}^{\mathrm{CH}(\mathcal{F})}(p_i^*)$
(7) $\quad\quad\quad \mathbf{q} \leftarrow \mathbf{R}\mathbf{q} + \mathbf{t}$
(8) $\quad\quad\quad$ **if** $\mathbf{d}_i^T \mathbf{q} > d^*$
(9) $\quad\quad\quad\quad p^* \leftarrow q$
(10) $\quad\quad\quad\quad d^* \leftarrow \mathbf{d}_i^T q$
(11) $\quad$ **until** $p^* = p_i^*$
(12) $\quad \overline{d}_i \leftarrow d^*$
(13) **return** $\mathrm{FDH}_n(\mathbf{D}, \overline{\mathbf{d}})$

FDH_n der transformierten Flächenmenge $\mathcal{F}$, sondern eine *Approximation* dieses Hüllkörpers. Nach jedem Bewegungsschritt wird die beim Hierarchieaufbau berechnete Fixed Directions Hull $F^0 = \mathrm{FDH}_n^0(\mathcal{F})$ entsprechend der Objektbewegung transformiert, was zu einer Fixed Directions Hull führt, deren begrenzende Halbebenen andere Richtungen als die durch D vorgegebenen aufweisen. Deshalb sind wir gezwungen, die auf D bezogene FDH_n der transformierten, ursprünglichen Fixed Directions Hull zu bestimmen.

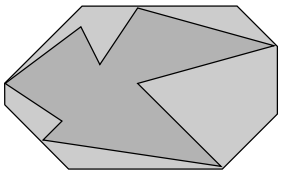
$$\mathrm{FDH}_n\big(\tau(\mathcal{F})\big) = \mathrm{FDH}_n\big(\tau(\mathcal{F}^0)\big) .$$

Der so entstandene Hüllkörper $F = \mathrm{FDH}_n(\tau(F^0))$ stellt im allgemeinen keine volumenminimale FDH_n der transformierten Flächenmenge dar, wie Abbildung 3.27 veranschaulicht. Für die Abstandsberechnung bedeutet diese Tatsache, daß wir als Abstand von F zu einer anderen FDH_n eine *untere Schranke* für die Distanz der $\mathrm{FDH}_n\big(\tau(\mathcal{F})\big)$ zu dieser anderen Fixed Directions Hull erhalten. Somit stellt der Hüllkörperabstand weiterhin eine konservative Abschätzung des Abstands der eingeschlossenen Flächenmengen dar.

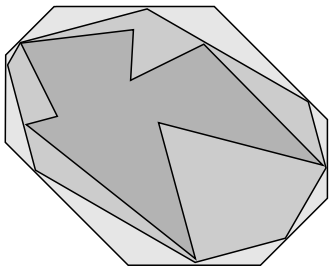
Ist die Zahl n der Richtungsvektoren, die den Hüllkörpertyp definieren, gering, so können wir die ursprüngliche Fixed Directions Hull mit der *Brute-Force-Methode* aktualisieren und anschließend die FDH_n der transformierten Eckpunkte berechnen.

Diese Vorgehensweise erfordert somit die Speicherung der Eckpunkte der beim Hierarchieaufbau berechneten $\mathrm{FDH}_n^0(\mathcal{F})$. Diese Punkte erhalten wir, indem wir den

Abbildung 3.27 Die Anwendung der Bewegungsabbildung auf die FDH_n des Objektes in seiner Ausgangslage liefert eine konservative Approximation der FDH_n des transformierten Objektes.



FDH_n des Objektes in Ausgangslage.



Tatsächliche und approximierte FDH_n nach der Objekttransformation.

Schnitt der n Halbebenen berechnen, die die FDH_n definieren. Dazu geht man zu einer *dualen* Darstellung der $FDH_n^0(\mathcal{F})$ über, in der jede begrenzende Ebene einem Punkt des $\mathbb{R}^3$ entspricht. Aus der konvexen Hülle dieser Punktmenge kann der *Schnitt der Halbebenen* einfach bestimmt werden. Im Rahmen der Rücktransformation ist es lediglich notwendig, die Flächen der konvexen Hülle als die gesuchten Eckpunkte des Halbraumschnitts zu interpretieren.

Der Vorteil dieser Aktualisierungsmethode gegenüber dem zuvor vorgestellten Verfahren besteht in deutlich *geringerem Rechenaufwand und Platzbedarf*.

Das Dualitätsverfahren

Erhöht man die Zahl der Richtungen in D, um die Approxmationsgüte der Fixed Directions Hulls zu verbessern, so steigt die Zahl der Eckpunkte drastisch an. Dies bedeutet nicht nur einen gestiegenen Platzbedarf zur Speicherung der Randrepräsentation der ursprünglichen FDH_n, sondern vor allem einen erhöhten Aktualisierungsaufwand. Das Brute-Force-Verfahren zur Transformation der ursprünglichen Fixed Directions Hull wird somit sehr schnell ineffizient, so daß wir uns nach alternativen Methoden zur Berechnung der approximierten FDH_n umsehen müssen.

Sei $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ die Bewegungsabbildung und sei $F^0 = FDH_n(\mathbf{D}, \overline{\mathbf{d}}^0)$ die $\mathcal{F}$ einschließende Fixed Directions Hull zum Zeitpunkt des Hierarchieaufbaus, dann besteht unsere Aufgabe darin, eine FDH_n für die Punktmenge $\tau(F^0) = \left\{\mathbf{R}\mathbf{x} + \mathbf{t} \,|\, \mathbf{D}\mathbf{x} \leq \overline{\mathbf{d}}^0\right\}$ zu finden. Die gesuchte FDH_n ist gegeben durch den Vektor $\overline{d}$ mit

$$\begin{aligned} \overline{d}_i &= \max\left\{\mathbf{d}_i^T(\mathbf{R}\mathbf{x} + \mathbf{t}) \,\middle|\, \mathbf{D}\mathbf{x} \leq \overline{\mathbf{d}}^0\right\} \\ &= \mathbf{d}_i^T\mathbf{t} + \max\left\{(\mathbf{d}_i^T\mathbf{R})\mathbf{x} \,|\, \mathbf{D}\mathbf{x} \leq \overline{\mathbf{d}}^0\right\} \\ &= b_i + \max\left\{\mathbf{c}_i^T\mathbf{x} \,|\, \mathbf{D}\mathbf{x} \leq \overline{\mathbf{d}}^0\right\}, \quad 1 \leq i \leq n, \end{aligned} \tag{3.14}$$

wobei b_i die i-te Komponente des Vektors $\mathbf{b} := \mathbf{D}\mathbf{t}$ und $\mathbf{c}_i$ den i-ten Zeilenvektor der

Matrix $\mathbf{C} := \mathbf{D}^T\mathbf{R}$ bezeichnet.
$\mathbf{C}$ und b sind somit unabhängig von dem zu aktualisierenden Hüllkörper und können nach einer Objektbewegung für alle FDH_n der Hierarchie vorberechnet werden. Die Lösung des Optimierungsproblems wird in [Kon98] mit Hilfe einer *Dualitätsbetrachtung* angegangen. Das duale Problem zu der in 3.14 formulierten Maximierungsaufgabe lautet:

$$\min\left\{\overline{\mathbf{d}}^{0T}\mathbf{y} \,\middle|\, \mathbf{D}^T\mathbf{y} = \mathbf{c}_i,\ \mathbf{y} \geq \mathbf{0},\ \overline{\mathbf{d}}^0, \mathbf{y} \in \mathbb{R}^n, \mathbf{D} \in \mathbb{R}^{n\times 3}, \mathbf{c}_i \in \mathbb{R}^3\right\}. \tag{3.15}$$

Da die Menge der primal zulässigen Lösungen $P = \{\mathbf{x} \in \mathbb{R}^3 \,|\, \mathbf{D}\mathbf{x} \leq \overline{\mathbf{d}}^0\}$ nichtleer und beschränkt ist, folgt aus dem Dualitätstheorem der Linearen Programmierung, daß die Menge der dual zulässigen Lösungen $Q = \{\mathbf{y} \in \mathbb{R}^n \,|\, \mathbf{D}^T\mathbf{y} = \mathbf{c}_i,\ \mathbf{y} \geq \mathbf{0}\}$ ebenfalls diese Eigenschaft besitzt. Der Vorteil der dualen Darstellung ist, daß die Menge Q unabhängig von der zu aktualisierenden FDH ist und die Eckpunktmenge des durch Q definierten Polyeders für die gesamte Approximationshierarchie vorberechnet werden kann. Da die Lösung des Optimierungsproblems in einem Eckpunkt angenommen wird, besteht die Aufgabe darin, den Eckpunkt mit minimalem Zielfunktionswert zu bestimmen. In [Kon98] wird gezeigt, daß die Zahl der dualen Eckpunkte bei einer geschickten Wahl der Richtungsvektoren aus D sehr moderat ansteigt. Im Fall der FDH_{14} sind beispielsweise lediglich zwei Eckpunkte zu betrachten.

Aus der Theorie der Linearen Programmierung wissen wir, daß die zulässigen Basislösungen von 3.15 gerade die Eckpunkte von Q sind. Da eine Basis von 3.15 einer Auswahl von drei linear unabhängigen Spaltenvektoren der Matrix $\mathbf{D}^T$ entspricht, hat jeder Eckpunkt von Q höchstens drei positive Koordinaten. Es genügt somit, lediglich den Wert der Basisvariablen zu speichern, da der Zielfunktionswert durch die übrigen Komponenten der Basislösung $\mathbf{y}$ nicht beeinflußt wird.

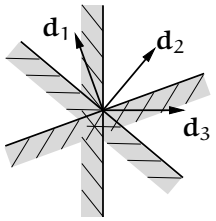
Jeder Eckpunkt des primalen Problems wird definiert durch den Schnitt von drei unterstützenden Ebenen mit linear unabhängigen Normalenvektoren. Ein solcher Punkt entspricht also einer Basis von 3.14, die durch die Auswahl von drei linear unabhängigen Zeilenvektoren (Richtungsvektoren) aus $\mathbf{D}$ charakterisiert ist. Die Indexmenge $I = \{j_1, j_2, j_3\}$ kennzeichnet die Auswahl. Da sich mehr als drei unterstützende Ebenen in einem Eckpunkt schneiden können, muß die zugehörige Basis nicht eindeutig sein. Wir interessieren uns jedoch nur für den Zielfunktionswert der entsprechenden Basislösung, so daß es keine Rolle spielt, welche Basis zu dessen Bestimmung herangezogen wird. Da die primalen Basen mit Eckpunkten des dualen Problems korrespondieren, sollten wir die Eckpunkte von Q, die den redundanten Basen von 3.14 entsprechen, ignorieren. Abbildung 3.28 zeigt für den zweidimensionalen Fall einen Eckpunkt von P, in dem sich drei Ebenen schneiden.

Wir wollen nun ein hinreichendes Kriterium zur Bestimmung redundanter Basen herleiten.

Lemma 3.12. *Sind* j_i, $0 \leq i \leq l$ *Indizes und* α_i, $1 \leq i \leq l \leq n$ *nichtnegative Koeffizienten, so daß für die Fixed Directions Hull* $F = FDH_n(\mathbf{D}, \overline{\mathbf{d}})$ *die folgende Bedingung erfüllt ist:*

$$\mathbf{d}_{j_0} = \sum_{i=1}^{l} \alpha_i \mathbf{d}_{j_i}.$$

Abbildung 3.28 *Drei* Halbebenen schneiden sich in einer Ecke des Polyeders:
$\exists \alpha_1, \alpha_3 \geq 0 : \mathbf{d}_2 = \alpha_1 \mathbf{d}_1 + \alpha_3 \mathbf{d}_3$ und $\forall \alpha_i, \alpha_j \geq 0 : \mathbf{d}_i \neq \alpha_2 \mathbf{d}_2 + \alpha_j \mathbf{d}_j$, $i \neq j \in \{1, 3\}$



Dann gilt:

$$\overline{d}_{j_0} \leq \sum_{i=1}^{l} \alpha_i \overline{d}_{j_i} .$$

Beweis. Die unterstützende Ebene $\Sigma(\mathbf{d}_{j_0}) = \{\boldsymbol{x} \mid \mathbf{d}_{j_0}^T \boldsymbol{x} = \overline{d}_{j_0}\}$ muß die FDH_n in einem Eckpunkt v schneiden. Für diesen Punkt gilt somit: $\mathbf{d}_i^T \boldsymbol{v} \leq \overline{d}_i$, $i = 1, \ldots, n$.
Da α_i nichtnegativ ist, erhalten wir:

$$\alpha_i \mathbf{d}_{j_i}^T \boldsymbol{v} \leq \alpha_i \overline{d}_{j_i} , \quad i = 1, \ldots, l .$$

Berücksichtigen wir nun die Annahme unseres Lemmas, so ergibt sich:

$$\sum_{i=1}^{l} \alpha_i \overline{d}_{j_i} \geq \sum_{i=1}^{l} \alpha_i \mathbf{d}_{j_i}^T \boldsymbol{v} = \sum_{i=1}^{l} (\alpha_i \mathbf{d}_{j_i})^T \boldsymbol{v} = \Big(\sum_{i=1}^{l} \alpha_i \mathbf{d}_{j_i} \Big)^T \boldsymbol{v} = \mathbf{d}_{j_0}^T \boldsymbol{v} = \overline{d}_{j_0} ,$$

was zu zeigen war. □

Abbildung 3.29 zeigt für $l = 2$ die Fälle $\overline{d}_{j_0} < \sum_{i=1}^{l} \overline{d}_{j_i}$ (a), $\overline{d}_{j_0} = \sum_{i=1}^{l} \overline{d}_{j_i}$ (b) sowie den unmöglichen Fall $\overline{d}_{j_0} > \sum_{i=1}^{l} \overline{d}_{j_i}$ (c) im $\mathbb{R}^2$.

Lemma 3.13. *Ist v ein Eckpunkt von P und $I_B = \{j_1, j_2, j_3\}$ eine Basis von 3.14, dann ist I_B eine redundante Basis, falls es einen Index j_0 und nichtnegative Koeffizienten α_i, $1 \leq i \leq 3$, gibt, so daß gilt:*

$$\mathbf{d}_{j_0} = \sum_{i=1}^{3} \alpha_i \mathbf{d}_{j_i} .$$

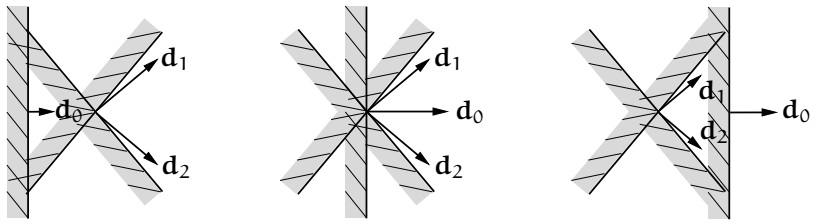
Beweis. Da I_B drei unterstützende Ebenen von P charakterisiert, die den Eckpunkt v definieren, gilt:

$$\mathbf{d}_{j_i}^T \boldsymbol{v} = \overline{d}_{j_i} , \quad 1 \leq i \leq 3 .$$

Wir zeigen zunächst, daß v innerhalb der durch $\mathbf{d}_{j_0}$ beschriebenen, unterstützenden Ebene von P liegt.

$$\mathbf{d}_{j_0}^T \boldsymbol{v} = \sum_{i=1}^{3} \alpha_i \mathbf{d}_{j_i}^T \boldsymbol{v} = \sum_{i=1}^{3} \alpha_i \overline{d}_{j_i} \geq \overline{d}_{j_0} .$$

Abbildung 3.29 Redundante, nichtredundante und unmögliche Begrenzungsebenen.



Aufgrund von Lemma 3.13 erhalten wir $\mathbf{d}_{j_0}^T \boldsymbol{v} \geq \overline{d}_{j_0}$. Somit folgt aus $v \in P$, daß $\mathbf{d}_{j_0}^T \boldsymbol{v} \leq \overline{d}_{j_0}$ und schließlich $\mathbf{d}_{j_0}^T \boldsymbol{v} = \overline{d}_{j_0}$ gilt.
Wir können somit einen Index aus I_B durch j_0 ersetzen, ohne daß die neue Basis einen anderen Eckpunkt als v definiert. I_B ist also eine redundante Basis. □

Es ist offensichtlich, daß für eine endliche Menge von mindestens drei nichtparallelen (Richtungs)vektoren gilt, daß mindestens eine dreielementige Teilmenge keinen der verbleibenden Vektoren durch eine nichtnegative Linearkombination darstellen kann. Daraus folgt, daß nicht alle einen Eckpunkt charakterisierenden Basen als redundant klassifiziert werden. In Abbildung 3.28 ist weder für $\mathbf{d}_1$ noch für $\mathbf{d}_3$ eine Darstellung als nichtnegative Linearkombination möglich, so daß nur die Basis $\{1, 3\}$ als redundant identifiziert wird.

Eine alternative Formulierung des Kriteriums aus Lemma 3.13 erhalten wir mit Hilfe der folgenden Beobachtung. Ist $\mathbf{B} = [\mathbf{d}_{j_1}, \mathbf{d}_{j_2}, \mathbf{d}_{j_3}]$ die durch die Basis $I_B = \{j_1, j_2, j_3\}$ definierte Matrix von linear unabhängigen Spaltenvektoren von $\mathbf{D}^T$ und $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)^T$ der Koeffizientenvektor der Linearkombination. Dann gilt:

$$\mathbf{d}_{j_0} = \sum_{i=1}^{3} \alpha_i \mathbf{d}_{j_i} \qquad \Longleftrightarrow \qquad \mathbf{B}\boldsymbol{\alpha} = \mathbf{d}_{j_0}\,, \quad j_0 \in I_B\,.$$

Die Basis I_B ist somit redundant, wenn gilt:

$$\exists j_0 \notin I_B : \boldsymbol{\alpha} = \mathbf{B}^{-1}\mathbf{d}_{j_0} \geq \mathbf{0}\,.$$

Wir können nun ein Verfahren angeben, das die Eckpunkte des dualen Polyeders Q berechnet. Dazu zählen wir alle dreielementigen Teilmengen $I_B = \{j_1, j_2, j_3\}$ von $\{1, \ldots, n\}$ auf und überprüfen, ob I_B eine Basis des primalen und somit des dualen Problems darstellt. Dies ist genau dann der Fall, wenn $\mathbf{B} = [\mathbf{d}_{j_1}, \mathbf{d}_{j_2}, \mathbf{d}_{j_3}]$ invertierbar ist. Haben wir eine Basis I_B gefunden, so überprüfen wir, ob diese Basis als redundant klassifiziert werden kann. Ist dies nicht möglich, so speichern wir die positiven Komponenten der Eckpunkte des dualen Polyeders. Diese erhalten wir entsprechend der Theorie der Linearen Programmierung als $\boldsymbol{y} = \mathbf{B}^{-1}\mathbf{c}_i$, mit $\boldsymbol{y} \geq \mathbf{0}$. Algorithmus 17 formalisiert die gerade vorgestellte Vorgehensweise.

Algorithmus 17 Ein Algorithmus zur Berechnung der dualen Eckpunkte des FDH-Approximationsproblems.

Eingabe: Die zu aktualisierende Fixed Directions Hull $F = FDH_n(\mathbf{D}, \overline{\mathbf{d}})$ und die Bewegungsabbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$.
Ausgabe: Die Eckpunktlisten $vlist[i]$, $1 \leq i \leq n$ der dualen Approximationsprobleme.

```
DUALVERTICES(F, R, t)
(1)   foreach I_B = {j1, j2, j3} ⊆ {1, ..., n}
(2)     B ← [d_j1, d_j2, d_j3]
(3)     if det(B) ≠ 0
(4)       RED ← false
(5)       foreach j0 ∈ {1, ..., n} \ I_B
(6)         if B^-1 d_j0 ≥ 0
(7)           RED ← true
(8)       if not RED
(9)         for i ← 1 to n
(10)          y ← B^-1 c_i
(11)          if y ≥ 0
(12)            vlist[i].APPEND([j1, j2, j3, y])
(13)  return vlist
```

Abstandsberechnung

Der quadratische Euklidische Abstand zweier Fixed Directions Hulls $F_i = FDH_n(\mathbf{D}, \overline{\mathbf{d}}_i)$, $i = 1, 2$ ergibt sich unter Berücksichtigung der Aktualisierungsabbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x}+\mathbf{t}$ als

$$\begin{aligned}
\delta^2(\tau(F_1), F_2) &= \min\left\{\|\mathbf{R}\mathbf{x}_1 + \mathbf{t} - \mathbf{x}_2\|^2 \,\middle|\, \mathbf{D}\mathbf{x}_1 \leq \overline{\mathbf{d}}_1,\ \mathbf{D}\mathbf{x}_2 \leq \overline{\mathbf{d}}_2\right\} \\
&= \min\left\{(\mathbf{R}\mathbf{x}_1 - \mathbf{x}_2)^2 \,\middle|\, \mathbf{D}\mathbf{x}_1 \leq \overline{\mathbf{d}}_1,\ \mathbf{D}(\mathbf{x}_2 + \mathbf{t}) \leq \overline{\mathbf{d}}_2\right\} \\
&= \min\left\{(\mathbf{R}\mathbf{x}_1 - \mathbf{x}_2)^2 \,\middle|\, \mathbf{D}\mathbf{x}_1 \leq \overline{\mathbf{d}}_1,\ \mathbf{D}\mathbf{x}_2 \leq \overline{\mathbf{d}}_2 - \mathbf{D}\mathbf{t}\right\}.
\end{aligned}$$

Somit erhalten wir auch bei diesem Hüllkörpertyp ein Optimierungsproblem mit quadratischer Zielfunktion und linearen Nebenbedingungen. Wie im Quaderfall wollen wir aus den in 3.7.4 genannten Gründen auf den Einsatz von Algorithmen der nichtlinearen Optimierung verzichten.

Ist die Zahl vorgegebener Richtungen groß, so werden die in 3.2 vorgestellten Verfahren zur Abstandsberechnung konvexer Polyeder interessant. Für die von uns eingesetzten FDH_{14}-Hüllkörper sind diese Algorithmen allerdings immer noch zu ineffizient.

Eine Beschleunigung des naiven Verfahrens durch Kantenklassifikation bezüglich einer hüllkörperinduzierten Unterteilung des Raums in Regionen, wie wir es im Quaderfall gesehen haben, ist hier nicht ohne weiteres möglich, da die Topologie der FDH_n-

Instanzen variieren kann und eine sinnvolle Raumpartitionierung nicht offensichtlich ist.

In den Arbeiten von [KZ97] und [Kon98] wird daher eine alternative Vorgehensweise vorgeschlagen. Verzichtet man auf die exakte Berechnung des Abstands und ermittelt eine *untere Schranke* für die Distanz der Fixed Directions Hulls, so stellt auch dieser Wert eine konservative Schranke für den Abstand der eingeschlossenen Flächenmenge dar. Um eine einfach zu berechnende Abstandsschranke für FDHs zu erhalten, wechseln die Autoren die der Abstandsberechnung zugrundeliegende Norm. Statt der Euklidischen Norm verwenden sie die sogenannte D-*induzierte Norm*, um die Distanz zweier FDHs zu berechnen.

Definition 3.6 (D-induzierte Norm). Sei $D \subseteq \mathbb{R}^3$ eine endliche Menge von Richtungsvektoren des $\mathbb{R}^3$, so daß die Menge $M := \{\mathbf{x} \in \mathbb{R}^3 \,|\, \forall \mathbf{d} \in D : \mathbf{d}^T\mathbf{x} \leq 1\}$ beschränkt ist. Dann lautet die D-induzierte Norm eines Vektors $\mathbf{x} \in \mathbb{R}^3$:

$$\|\mathbf{x}\|_D := \max_{\mathbf{d} \in D} \mathbf{d}^T\mathbf{x} \,.$$

□

Das folgende Lemma rechtfertigt die Wahl dieser Norm als Grundlage der Abstandsberechnung.

Lemma 3.14. *Es existiert eine reelle Konstante* $c > 0$*, so daß für alle* $\mathbf{x} \in \mathbb{R}^3$ *gilt:*

$$\|\mathbf{x}\| \geq \|\mathbf{x}\|_D \geq c\|\mathbf{x}\| \,.$$

Beweis. Die Euklidische Norm stellt eine obere Schranke für die D-induzierte Norm dar, da für alle $\mathbf{d} \in D$ gilt:

$$\mathbf{d}^T\mathbf{x} \leq \left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)^T \mathbf{x} = \sqrt{\mathbf{x}^T\mathbf{x}} = \|\mathbf{x}\|$$

und somit aus der Monotonie von max folgt:

$$\|\mathbf{x}\|_D = \max_{\mathbf{d} \in D} \mathbf{d}^T\mathbf{x} \leq \|\mathbf{x}\| \,.$$

Für die zweite Ungleichung betrachten wir das Bild M des Einheitskugelrandes $\partial S^0 = \left\{\mathbf{x} \in \mathbb{R}^3 \,|\, \|\mathbf{x}\| = 1\right\}$ unter der D-induzierten Norm, d.h.

$$M = \left\{\|\mathbf{x}\|_D \,\middle|\, \|\mathbf{x}\| = 1\right\} \,.$$

Da diese Punktmenge kompakt und $\|.\|_D$ eine stetige Abbildung ist, muß M ebenfalls kompakt sein. Für jeden Punkt $p \in \partial S^0$ gilt $\mathbf{p} \neq \mathbf{0}$ und somit $\|\mathbf{p}\| > 0$.
Wir wählen nun $c := \inf M = \min M$, woraus $c > 0$ folgt. Somit erhalten wir für einen beliebigen Vektor $\mathbf{x} = \|\mathbf{x}\|\boldsymbol{e} \in \mathbb{R}^3$, $\|\boldsymbol{e}\| = 1$:

$$\|\mathbf{x}\|_D = \|\mathbf{x}\|\|\boldsymbol{e}\|_D \geq c\|\mathbf{x}\| \,.$$

□

Wir wollen nun analog zu unserer Vorgehensweise in 3.1 den D-induzierten Abstand zweier Punktmengen als Distanz der nahesten Punkte definieren.

Definition 3.7 (D-induzierter Abstand). Sind $X_1, X_2 \subseteq \mathbb{R}^3$ zwei nichtleere Mengen und $\|.\|_D$ die D-induzierte Norm im $\mathbb{R}^3$, so verstehen wir unter dem D-induzierten Abstand δ_D von X_1 und X_2 den Wert:

$$\delta_D(X_1, X_2) := \min_{\boldsymbol{x}_1 \in X_1, \boldsymbol{x}_2 \in X_2} \|x_1 - x_2\|_D \,.$$

□

Um den D-induzierten Abstand zwischen X_1 und X_2 zu berechnen, müssen wir das folgende Optimierungsproblem lösen:

$$\min \left\{ \max_{\mathbf{d} \in D} \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \,|\, \boldsymbol{x}_1 \in X_1,\, \boldsymbol{x}_2 \in X_2 \right\} .$$

Den optimalen Zielfunktionswert $\delta_D(X_1, X_2)$ erhält man als Lösung des folgenden Programms:

$$\begin{array}{ll} \min & \delta_D \\ \text{s.d.} & \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \geq \delta_D \quad \forall \mathbf{d} \in D \\ & \boldsymbol{x}_1 \in X_2,\, \boldsymbol{x}_2 \in X_2,\, \delta_D \geq 0 \end{array} \tag{3.16}$$

(3.17)

Im Fall der Fixed Directions Hulls sind wir also gezwungen, ein Lineares Programm in 3 Variablen und $3n + 1$ Ungleichungen zu lösen. Da die Lösung dieses Problems mit Methoden der Linearen Programmierung im Rahmen unseres Branch-and-Bound-Verfahrens zu aufwendig ist, wollen wir das Optimierungsproblem 3.16 weiter vereinfachen.

$$\begin{aligned} &\min \left\{ \max_{\mathbf{d} \in D} \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \,\middle|\, \boldsymbol{x}_1 \in X_1,\, \boldsymbol{x}_2 \in X_2 \right\} \geq \\ &\min \left\{ \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \,\middle|\, \boldsymbol{x}_1 \in X_1,\, \boldsymbol{x}_2 \in X_2 \right\} \quad \forall \mathbf{d} \in D \,. \end{aligned}$$

Somit gilt im speziellen:

$$\min \left\{ \max_{\mathbf{d} \in D} \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \,\middle|\, \boldsymbol{x}_1 \in X_1,\, \boldsymbol{x}_2 \in X_2 \right\} \geq \max_{\mathbf{d} \in D} \min \left\{ \mathbf{d}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \,\middle|\, \boldsymbol{x}_1 \in X_1,\, \boldsymbol{x}_2 \in X_2 \right\}.$$

Diese untere Schranke für den D-induzierten Abstand zweier Punktmengen wollen wir als Abstandsmaß unserer Distanzberechnung zugrunde legen. Handelt es sich bei den Punktmengen um Fixed Directions Hulls $F_i = FDH(\mathbf{D}, \overline{\mathbf{d}}^i)$, so kann die Schranke auf

einfache Weise ermittelt werden:

$$\begin{aligned}
&\max_{\mathbf{d}\in D} \min \left\{ \mathbf{d}^T(\mathbf{x}_1 - \mathbf{x}_2) \,\middle|\, \mathbf{x}_1 \in F_1,\, \mathbf{x}_2 \in F_2 \right\} \\
&= \max_{\mathbf{d}\in D} \Big(\min \left\{ \mathbf{d}^T\mathbf{x}_1 \,\middle|\, \mathbf{x}_1 \in F_1 \right\} - \max \left\{ \mathbf{d}^T\mathbf{x}_2 \,\middle|\, \mathbf{x}_2 \in F_2 \right\} \Big) \\
&= \max_{\mathbf{d}\in D} -\Big(\max \left\{ -\mathbf{d}^T\mathbf{x}_1 \,\middle|\, \mathbf{x}_1 \in F_1 \right\} + \max \left\{ \mathbf{d}^T\mathbf{x}_2 \,\middle|\, \mathbf{x}_2 \in F_2 \right\} \Big) \\
&= \max_{1\le i\le n} \left\{ -\left(\overline{d}^1_{2i-1} + \overline{d}^2_{2i}\right),\, -\left(\overline{d}^1_{2i} + \overline{d}^2_{2i-1}\right) \right\} \\
&= -\min_{1\le i\le n} \left\{ \overline{d}^1_{2i-1} + \overline{d}^2_{2i},\, \overline{d}^1_{2i} + \overline{d}^2_{2i-1} \right\}. \qquad (3.18)
\end{aligned}$$

Dabei haben wir verwendet, daß für jedes Paar von Richtungsvektoren $\mathbf{d}_{2i-1}$ und $\mathbf{d}_{2i}$, $1 \le i \le n$ gilt: $\mathbf{d}_{2i-1} = -\mathbf{d}_{2i}$. Der Euklidische Abstand zweier FDHs kann somit folgendermaßen nach unten abgeschätzt werden:

$$\delta(F_1, F_2) \ge \delta_D(F_1, F_2) \ge -\min_{1\le i\le n} \left\{ \overline{d}^1_{2i-1} + \overline{d}^2_{2i},\, \overline{d}^1_{2i} + \overline{d}^2_{2i-1} \right\}.$$

Der Beweis von Lemma 3.14 hat gezeigt, daß die Güte der Approximation im wesentlichen von den folgenden Parametern abhängt:

- der Zahl der Richtungsvektoren **und**
- der Wahl der Richtungsvektoren.

Mit Hilfe der eingeführten Konstante c läßt sich der größte relative Fehler bestimmen, der bei der Abstandsberechnung durch den Übergang von der Euklidischen Norm zur D-induzierten Norm gemacht werden kann:

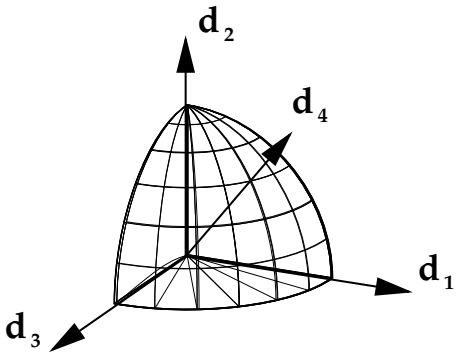
$$e_D = \frac{\|x\| - \|x\|_D}{\|x\|} \le 1 - c\,.$$

Da der Beweis von Lemma 3.14 konstruktiv war, können wir c für eine gegebene Menge von Richtungsvektoren D mit Hilfe von Verfahren der nichtlinearen Programmierung bestimmen. Hierzu müssen wir das folgende Optimierungsproblem lösen:

$$\begin{aligned}
&\min \ \max_{1\le i\le n} \sum_{j=1}^{3} d_{ij}x_j \\
&\text{s.d.} \sum_{j=1}^{3} x_j^2 = 1 \\
&\mathbf{x} \in \mathbb{R}^3
\end{aligned}
\quad\Longleftrightarrow\quad
\begin{aligned}
&\min \ c \\
&\text{s.d.} \sum_{j=1}^{3} x_j^2 = 1 \\
&\sum_{j=1}^{3} d_{ij}x_j \ge c\,, \quad 1 \le i \le n \\
&\mathbf{x} \in \mathbb{R}^3
\end{aligned}$$

Im Fall der von uns betrachteten FDH_{14}-Hüllkörper läßt sich das Optimierungsproblem über seine geometrische Anschauung lösen.

Abbildung 3.30 Einschränkung der Problemstellung auf einen Oktanten.



Satz 3.15. *Sind* $F_i = FDH_{14}(\mathbf{D}, \overline{\mathbf{d}}_i)$, $i = 1, 2$ *zwei* FDH_{14} *Hüllkörper, dann beträgt der relative Fehler bei der konservativen Abschätzung ihres Euklidischen Abstands durch den* D*-induzierten Abstand höchstens* $e_D = 1 - c = 1 - 1/\sqrt{5 - 2\sqrt{3}} \approx 0,193$*, d.h. es gilt*

$$\delta(F_1, F_2) \geq \delta_D(F_1, F_2) \geq \frac{1}{\sqrt{5 - 2\sqrt{3}}} \delta(F_1, F_2)$$

Beweis. Lemma 3.14 hat gezeigt, daß eine reelle Konstante $c > 0$ existiert, für die gilt:

$$c \leq \frac{\|x\|_D}{\|x\|} \leq 1 \quad \forall \boldsymbol{x} \in \mathbb{R}^3 .$$

Um die Konstante c zu bestimmen, muß man für jeden Punkt des Einheitskugelrands ∂S^0 die längste Projektion auf die Richtungsvektoren aus D betrachten. Das Minimum dieser Menge liefert den gesuchten Wert:

$$c = \min \left\{ \max_{\mathbf{d} \in D} \mathbf{d}^T \boldsymbol{x} \,\middle|\, \boldsymbol{x} \in \mathbb{R}^3, \|\boldsymbol{x}\| = 1 \right\} .$$

Im Fall der FDH_{14} sind die zu berücksichtigenden Richtungen durch die Koordinatenachsen und die Raumdiagonalen vorgegeben. Aufgrund dieser Symmetrie genügt es, sich auf einen der acht Oktanten zu beschränken. Abbildung 3.30 beschreibt die geometrische Anschauung der Problemstellung. Im folgenden werden wir daher lediglich den durch die positiven Koordinatenachsen bestimmten Oktanten betrachten. Wir suchen also einen Punkt aus ∂S^0, der innerhalb dieses Oktanten liegt und dessen maximale Projektion auf die Richtungsvektoren aus D minimal ist. Offensichtlich sind daher lediglich die Vektoren $\mathbf{d}_1 = (1, 0, 0)^T$, $\mathbf{d}_2 = (0, 1, 0)^T$, $\mathbf{d}_3 = (0, 0, 1)^T$ und $\mathbf{d}_4 = 1/\sqrt{3}(1, 1, 1)^T$ interessant. Der relevante Ausschnitt des Einheitskugelrandes entspricht dem durch $\mathbf{d}_1$, $\mathbf{d}_2$ und $\mathbf{d}_3$ definierten sphärischen Dreieck Δ der Einheitskugel S^0 (Abbildung 3.30):

$$\Delta := \left\{ \boldsymbol{x} \in \mathbb{R}^3_{\geq 0} \,\middle|\, \|\boldsymbol{x}\| = 1 \right\} .$$

Für die Konstante c gilt somit:

$$c = \min\left\{ \max_{1\le i\le 4} \mathbf{d}_i^T \mathbf{x} \,\middle|\, \mathbf{x} \in \Delta \right\} .$$

Die geometrische Interpretation dieser Problemstellung ist die Suche nach einem Punkt aus Δ, dessen Projektion auf alle vier Richtungsvektoren identisch ist. Da alle beteiligten Vektoren Einheitslänge haben, handelt es sich dabei um einen Vektor, der zu allen vier Richtungsvektoren den gleichen Winkel einnimmt. Innerhalb des betrachteten Oktanten gibt es jedoch lediglich einen Vektor, dessen Projektion auf drei Richtungsvektoren $\mathbf{d}_1$, $\mathbf{d}_2$ und $\mathbf{d}_3$ identisch ist. Dabei handelt es sich gerade um den vierten Richtungsvektor $\mathbf{d}_4$, dessen maximale Projektion 1 beträgt:

$$\max_{1\le i\le 4} \mathbf{d}_i^T \mathbf{d}_4 = 1 .$$

Betrachten wir drei andere Richtungsvektoren, wobei wir aus Symmetriegründen o.B.d.A. $\mathbf{d}_1$, $\mathbf{d}_2$ und $\mathbf{d}_4$ wählen, so liegt der Vektor, der bezüglich allen dreien die gleiche Projektion besitzt, außerhalb von Δ. Es gilt nämlich:

$$\begin{aligned}
&(1) \qquad \mathbf{d}_1^T \mathbf{x} = \mathbf{d}_3^T \mathbf{x} \quad &&\Longleftrightarrow\ x_1 = x_3 ,\\
&(2) \qquad \mathbf{d}_1^T \mathbf{x} = \mathbf{d}_4^T \mathbf{x} \quad &&\Longleftrightarrow\ x_1 = \frac{1}{\sqrt{3}}(x_1 + x_2 + x_3) ,\\
&(3) \qquad x_1^2 + x_2^2 + x_3^2 = 1 .
\end{aligned}$$

Einsetzen liefert den gesuchten Vektor $\mathbf{x} = (x_1, x_2, x_3)^T \notin \Delta$:

$$x_1 = \frac{1}{9 - 4\sqrt{3}} > 0 , \quad x_2 = \frac{\sqrt{3} - 2}{9 - 4\sqrt{3}} < 0 , \quad x_3 = \frac{1}{9 - 4\sqrt{3}} > 0 .$$

Somit können wir festhalten:

$$\begin{aligned}
c_1 &:= \min\left\{ \max_{1\le i\le 4} \mathbf{d}_i^T \mathbf{x} \,\middle|\, \exists\, i,j,k : 1 \le i < j < k \le 4 \ \wedge\ \mathbf{d}_i^T \mathbf{x} = \mathbf{d}_j^T \mathbf{x} = \mathbf{d}_k^T \mathbf{x},\ \mathbf{x} \in \Delta \right\}\\
&= \|\mathbf{x}^*\|_D = 1 ,
\end{aligned}$$

mit $\mathbf{x}^* := \mathbf{d}_4$.
Als nächstes bestimmen wir die Vektoren, die zu genau zwei der Vektoren aus D den gleichen Winkel annehmen.

1.Fall: $\mathbf{d}_1^T \mathbf{x} = \mathbf{d}_2^T \mathbf{x} \ \vee\ \mathbf{d}_1^T \mathbf{x} = \mathbf{d}_3^T \mathbf{x} \ \vee\ \mathbf{d}_2^T \mathbf{x} = \mathbf{d}_3^T \mathbf{x}$
Aus Symmetriegründen genügt es den Fall $\mathbf{d}_1^T \mathbf{x} = \mathbf{d}_3^T \mathbf{x}$ innerhalb des durch $\mathbf{d}_1$, $\mathbf{d}_2$ und $\mathbf{d}_4$ definierten Sphärischen Dreiecks zu untersuchen. Die Punkte aus Δ, die diese Eigenschaft erfüllen, können folgendermaßen beschrieben werden:

$$\mathbf{x}(\varepsilon) = \frac{1}{\sqrt{2 + \varepsilon^2}} (1, \varepsilon, 1)^T , \quad \varepsilon \in [0, 1] .$$

Als Projektionen dieser Punkte auf die Richtungsvektoren $\mathbf{d}_i$, $1 \leq i \leq 4$ erhalten wir:

$$\begin{aligned}\frac{1}{\sqrt{3}} \leq \mathbf{d}_1^T\mathbf{x} = \mathbf{d}_3^T\mathbf{x} &= \frac{1}{\sqrt{2+\varepsilon^2}} \leq \frac{1}{\sqrt{2}}\,,\\ 0 \leq \mathbf{d}_2^T\mathbf{x} &= \frac{\varepsilon}{\sqrt{2+\varepsilon^2}} \leq \frac{1}{\sqrt{3}}\,,\\ \frac{\sqrt{2}}{\sqrt{3}} \leq \mathbf{d}_4^T\mathbf{x} &= \frac{2+\varepsilon}{\sqrt{3}\sqrt{2+\varepsilon^2}} \leq 1\,.\end{aligned}$$

Hieraus folgt:

$$0 \leq \mathbf{d}_2^T\mathbf{x} \leq \mathbf{d}_1^T\mathbf{x} = \mathbf{d}_3^T\mathbf{x} < \frac{\sqrt{2}}{\sqrt{3}} \leq \mathbf{d}_4^T\mathbf{x}$$

und somit:

$$\begin{aligned}c_2 &:= \min\Big\{ \max_{1\leq i\leq 4} \mathbf{d}_i^T\mathbf{x} \,\Big|\, \exists\, i,j : 1 \leq i < j \leq 3 \wedge \mathbf{d}_i^T\mathbf{x} = \mathbf{d}_j^T\mathbf{x},\, \mathbf{x} \in \Delta\Big\}\\ &= \|\mathbf{x}^*\|_D = \frac{\sqrt{2}}{\sqrt{3}}\,,\end{aligned}$$

mit $\mathbf{x}^* = \mathbf{x}(0) = 1/\sqrt{2}(1,0,1)^T$.

2.Fall: $\mathbf{d}_1^T\mathbf{x} = \mathbf{d}_4^T\mathbf{x}$, $\vee$ $\mathbf{d}_2^T\mathbf{x} = \mathbf{d}_4^T\mathbf{x}$, $\vee$ $\mathbf{d}_3^T\mathbf{x} = \mathbf{d}_4^T\mathbf{x}$

Aus Symmetriegründen beschränken wir uns auf den Fall $\mathbf{d}_1^T\mathbf{x} = \mathbf{d}_4^T\mathbf{x}$.
Diese Bedingung liefert:

$$x_2 + x_3 = (\sqrt{3}-1)x_1\,. \tag{3.19}$$

Betrachtet man $(\mathbf{d}_1^T\mathbf{x})^2 = (\mathbf{d}_4^T\mathbf{x})^2$ so ergibt sich:

$$\begin{aligned}x_1^2 &= \frac{1}{3}(x_1+x_2+x_3)^2\\ &= \frac{1}{3}(x_1^2+x_2^2+x_3^2) + \frac{2}{3}(x_1x_2+x_1x_3+x_2x_3)\\ &= \frac{1}{3} + \frac{2}{3}x_1(x_2+x_3) + \frac{2}{3}x_2x_3\,.\end{aligned} \tag{3.20}$$

Kombiniert man 3.19 und 3.20, so erhält man:

$$\begin{aligned}&x_1^2 = \frac{1}{3} + \frac{2}{3}(\sqrt{3}-1)x_1^2 + \frac{2}{3}x_2x_3\\ \iff\quad &x_1^2 = \frac{1}{5-2\sqrt{3}} + \frac{2}{5-2\sqrt{3}}x_2x_3\,.\end{aligned}$$

1. Fall: $x_2x_3 = 0 \quad\iff\quad x_2 = 0 \vee x_3 = 0$
   Aus $x_2x_3 = 0$ ergibt sich:

$$(\mathbf{d}_1^T\mathbf{x})^2 = (\mathbf{d}_4^T\mathbf{x})^2 = \frac{1}{5-2\sqrt{3}}\,.$$

Falls $x_2 = 0$ gilt, folgt $(\mathbf{d}_2^T\mathbf{x})^2 = 0$ und schließlich:

$$(\mathbf{d}_3^T\mathbf{x})^2 = x_3^2 = 1 - x_1^2 = \frac{4-2\sqrt{3}}{5-2\sqrt{3}} < (\mathbf{d}_1^T\mathbf{x})^2 .$$

Somit erhalten wir:

$$0 = \mathbf{d}_2^T\mathbf{x} < \mathbf{d}_3^T\mathbf{x} < \frac{1}{\sqrt{5-2\sqrt{3}}} = \mathbf{d}_4^T\mathbf{x} = \mathbf{d}_1^T\mathbf{x} .$$

Der Fall $x_3 = 0$ liefert analog

$$(d_2^T x)^2 = \frac{4-2\sqrt{3}}{5-2\sqrt{3}} \quad \text{und} \quad (d_3^T x)^2 = 0 ,$$

woraus folgt:

$$0 = \mathbf{d}_3^T\mathbf{x} < \mathbf{d}_2^T\mathbf{x} < \frac{1}{\sqrt{5-2\sqrt{3}}} = \mathbf{d}_4^T\mathbf{x} = \mathbf{d}_1^T\mathbf{x} .$$

2. Fall: $x_2x_3 > 0$
   Aus $x_2x_3 > 0$ ergibt sich:

$$(\mathbf{d}_1^T\mathbf{x})^2 = (\mathbf{d}_4^T\mathbf{x})^2 > \frac{1}{5-2\sqrt{3}} .$$

   Der Vektor $\mathbf{x}$ kann nicht optimal sein, da

$$\max_{1\le i\le 4} \mathbf{d}_i^T\mathbf{x} > \frac{1}{\sqrt{5-2\sqrt{3}}} .$$

   gilt.

Abschließend erhalten wir also:

$$\begin{aligned} c_3 &:= \min\left\{ \max_{1\le i\le 4} \mathbf{d}_i^T\mathbf{x} \,\middle|\, \exists\, i : 1 \le i \le 3 \wedge \mathbf{d}_i^T\mathbf{x} = \mathbf{d}_4^T\mathbf{x},\, \mathbf{x} \in \Delta \right\} \\ &= \|\mathbf{x}^*\|_D = \frac{1}{\sqrt{5-2\sqrt{3}}} , \end{aligned}$$

mit $\mathbf{x}^* = \frac{1}{\sqrt{5-2\sqrt{3}}} \left(1, 0, \sqrt{4-2\sqrt{3}}\right)^T$.

Aus $c_1 = 1 > c_2 = \frac{\sqrt{2}}{\sqrt{3}} > c_3 = \frac{1}{\sqrt{5-2\sqrt{3}}}$ folgt schließlich unsere Behauptung:

$$\begin{aligned} c &= \min\left\{ \max_{\mathbf{d}\in D} \mathbf{d}^T\mathbf{x} \,\middle|\, \|\mathbf{x}\| = 1 \right\} \\ &= \min\{c_1, c_2, c_3\} = \frac{1}{\sqrt{5-2\sqrt{3}}} . \end{aligned}$$

□

Da wir zur Abstandsberechnung zweier FDHs den D-induzierten Abstand nach unten abschätzen, können wir letztendlich keine Garantie für die Güte der ermittelten Distanz geben Die Kosten der Abstandsberechnung zweier Fixed Directions Hulls $F_i = FDH_n(\mathbf{D}, \overline{\mathbf{d}}_i)$, $i = 1, 2$ betragen nach 3.18:

$$C_D(F_1, F_2) = 1\,Mult + 2n\,Add + (n-1)\,Comp$$

bzw. im Fall der von uns verwendeten FDH_{14}:

$$C_D(F_1, F_2) = 1\,Mult + 14\,Add + 13\,Comp\,.$$

3.7.6 Der Abstand zweier einschließender achsenorientierter Boxen

Aktualisierung

Da achsenorientierte Boxen minimalen Volumens als Spezialfall der Fixed Directions Hulls aufgefaßt werden können, lassen sich prinzipiell alle in Abschnitt 3.7.5 vorgestellten Aktualisierungsverfahren auch auf AABBs anwenden. Die achsenorientierte Ausrichtung der Iso-Box erlaubt es, die meisten dieser Algorithmen effizienter zu gestalten, als dies im Fall allgemeiner FDH_n möglich gewesen ist. Im folgenden wollen wir dieses Optimierungspotential für den Hill-Climbing-Algorithmus und das Approximationsverfahren erläutern.

Der Hill-Climbing-Algorithmus

Das Hill-Climbing-Verfahren aktualisiert sukzessive die Extrempunkte der Iso-Box entlang der sechs Richtungen, die die AABB beschreiben. Ausgehend von dem jeweiligen Extrempunkt des letzten Zeitschritts, wird der neue Punkt durch lokale Austauschoperationen bestimmt. Dazu wird untersucht, ob einer der Nachbarknoten eine größere Projektion auf die gerade betrachtete Richtung besitzt. Falls ja, ist dieser Nachbarknoten ein möglicher Kandidat für den aktuellen Extrempunkt entlang dieser Richtung.

Im Fall der achsenorientierten Box sind die Richtungen durch die Koordinatenachsen vorgegeben, so daß man die zu bestimmenden Projektionen als Koordinaten des betrachteten Punktes erhält. Ausgehend von einem Extrempunkt sucht man also auf dem Weg über die Nachbarknoten den Punkt mit minimaler bzw. maximaler x-, y- oder z-Koordinate. Der Optimalitätstest ist somit lediglich ein Koordinatenvergleich und erfordert keine Vektormultiplikation, wie im allgemeinen Fall.

Das Approximationsverfahren

Neben den Algorithmen, die die exakte FDH_n der transformierten Flächenmenge ermitteln, haben wir auch solche Verfahren betrachtet, die eine angenäherte FDH_n bestimmen. Dazu wird die Fixed Directions Hull dieser Flächenmenge in der Ausgangslage, d.h. beim Hierarchieaufbau entsprechend der Objektbewegung transformiert und für die so entstandende Punktmenge eine FDH_n berechnet. Bei dieser letzten FDH_n handelt es sich um einen Hüllkörper der transformierten Flächenmenge, der jedoch im allgemeinen nicht volumenminimal ist.

Im Fall einer achsenorientierten Box $A = AABB(\mathbf{d}^{min}, \mathbf{d}^{max})$ kann die approximierte Iso-Box $\tilde{A} = AABB(\tilde{\mathbf{d}}^{min}, \tilde{\mathbf{d}}^{max})$ unter Ausnutzung der Beziehung $\max\{a+b, a+c\} = a + \max\{b, c\}$ effizient berechnet werden. Sei $\tau : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$ die Bewegungsabbildung und bezeichnen wir mit $[\mathbf{v}]_i$ die i-te Komponente des Vektors $\mathbf{v}$, $1 \leq i \leq 3$, so gilt:

$$
\begin{aligned}
&\tilde{d}_i^{max} \\
&= \max\Big\{ \big[\mathbf{R}^T(d_1^{min}, d_2^{min}, d_3^{min})\big]_i,\ \big[\mathbf{R}^T(d_1^{max}, d_2^{min}, d_3^{min})\big]_i,\ \big[\mathbf{R}^T(d_1^{min}, d_2^{max}, d_3^{min})\big]_i, \\
&\qquad \big[\mathbf{R}^T(d_1^{min}, d_2^{min}, d_3^{max})\big]_i,\ \big[\mathbf{R}^T(d_1^{max}, d_2^{max}, d_3^{min})\big]_i,\ \big[\mathbf{R}^T(d_1^{max}, d_2^{min}, d_3^{max})\big]_i, \\
&\qquad \big[\mathbf{R}^T(d_1^{min}, d_2^{max}, d_3^{max})\big]_i,\ \big[\mathbf{R}^T(d_1^{max}, d_2^{max}, d_3^{max})\big]_i \Big\} + t_i \\
&= \max\big\{ R_{i1}d_1^{min} + R_{i2}d_2^{min} + R_{i3}d_3^{min}, R_{i1}d_1^{max} + R_{i2}d_2^{min} + R_{i3}d_3^{min}, \\
&\qquad R_{i1}d_1^{min} + R_{i2}d_2^{max} + R_{i3}d_3^{min}, R_{i1}d_1^{min} + R_{i2}d_2^{min} + R_{i3}d_3^{max}, \\
&\qquad R_{i1}d_1^{max} + R_{i2}d_2^{max} + R_{i3}d_3^{min}, R_{i1}d_1^{max} + R_{i2}d_2^{min} + R_{i3}d_3^{max}, \\
&\qquad R_{i1}d_1^{min} + R_{i2}d_2^{max} + R_{i3}d_3^{max}, R_{i1}d_1^{max} + R_{i2}d_2^{max} + R_{i3}d_3^{max}\big\} + t_i \\
&= \max\Big\{ R_{i1}d_1^{min} + \max\big\{ R_{i2}d_2^{min} + R_{i3}d_3^{min}, R_{i2}d_2^{max} + R_{i3}d_3^{min}, \\
&\qquad\qquad R_{i2}d_2^{min} + R_{i3}d_3^{max}, R_{i2}d_2^{max} + R_{i3}d_3^{max}\big\} \\
&\qquad R_{i1}d_1^{max} + \max\big\{ R_{i2}d_2^{min} + R_{i3}d_3^{min}, R_{i2}d_2^{max} + R_{i3}d_3^{min}, \\
&\qquad\qquad R_{i2}d_2^{min} + R_{i3}d_3^{max}, R_{i2}d_2^{max} + R_{i3}d_3^{max}\big\} \Big\} + t_i \\
&= \max\Big\{ R_{i2}d_2^{min} + \max\big\{ R_{i3}d_3^{min}, R_{i3}d_3^{max}\big\}, R_{i2}d_2^{max} + \max\big\{ R_{i3}d_3^{min}, R_{i3}d_3^{max}\big\} \Big\} \\
&\quad + \max\big\{ R_{i1}d_1^{min}, R_{i1}d_1^{max}\big\} + t_i \\
&= \max\big\{ R_{i1}d_1^{min}, R_{i1}d_1^{max}\big\} + \max\big\{ R_{i2}d_2^{min}, R_{i2}d_2^{max}\big\} + \max\big\{ R_{i3}d_3^{min}, R_{i3}d_3^{max}\big\} + t_i.
\end{aligned}
$$

Analog ergibt sich für $\tilde{d}_i^{min}$, $1 \leq i \leq 3$:

$$\tilde{d}_i^{min} = \min\big\{ R_{i1}d_1^{min}, R_{i1}d_1^{max}\big\} + \min\big\{ R_{i2}d_2^{min}, R_{i2}d_2^{max}\big\} + \min\big\{ R_{i3}d_3^{min}, R_{i3}d_3^{max}\big\} + t_i.$$

Diese Beobachtung erlaubt die Formulierung von Algorithmus 18 zur effizienten Aktualisierung einer Iso-Box mit Hilfe des Approximationsverfahrens. Die Kosten der Aktualisierung einer achsenorientierten Box $A = AABB(\mathbf{d}^{min}, \mathbf{d}^{max})$ betragen somit:

$$C_A(A) = 18\,\mathrm{Mult} + 18\,\mathrm{Add} + 9\,\mathrm{Comp}\,.$$

Abstandsberechnung

Die Abstandsberechnung zweier achsenorientierter Boxen gestaltet sich wesentlich einfacher als im Fall allgemeiner FDHs. Der Grund hierfür ist die Ausrichtung aller Seitenflächen entlang der Koordinatenachsen, so daß der Verbindungsvektor der nahesten

Algorithmus 18 Ein Algorithmus zur Aktualisierung einer achsenorientierten Box mit Hilfe des Approximationsverfahrens.

Eingabe: Die zu aktualisierende Iso-Box $A = AABB(\mathbf{d}^{min}, \mathbf{d}^{max})$ sowie die Bewegungsabbildung $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$.

Ausgabe: Die aktualisierte Iso-Box $\tilde{A} = AABB(\tilde{\mathbf{d}}^{min}, \tilde{\mathbf{d}}^{max})$.

UPDATEAPPROXAABB($A, \mathbf{R}, \mathbf{t}$)

```
(1)   for i ← 1 to 3
(2)     d̃_i^min ← t_i
(3)     d̃_i^max ← t_i
(4)     for j ← 1 to 3
(5)       a ← R_ij d_j^min
(6)       b ← R_ij d_j^max
(7)       if a < b
(8)         d̃_i^min ← d̃_i^min + a
(9)         d̃_i^max ← d̃_i^max + b
(10)      else
(11)        d̃_i^min ← d̃_i^min + b
(12)        d̃_i^max ← d̃_i^max + a
(13)  return AABB(d̃^min, d̃^max)
```

Punkte (Abstandsvektor) komponentenweise berechnet werden kann. Dazu projiziert man die beiden zu betrachtenden, achsenorientierten Boxen $A_i = AABB(\mathbf{d}_i^{min}, \mathbf{d}_i^{max})$, $i = 1, 2$ auf die einzelnen Koordinatenachsen. Die Box A_i, $i = 1, 2$ wird somit durch drei Koordinatenintervalle beschrieben:

$$X_i = [x_i^{min}, x_i^{max}]\,,\ Y_i = [y_i^{min}, y_i^{max}]\,,\ Z_i = [z_i^{min}, z_i^{max}], \quad i = 1, 2,$$

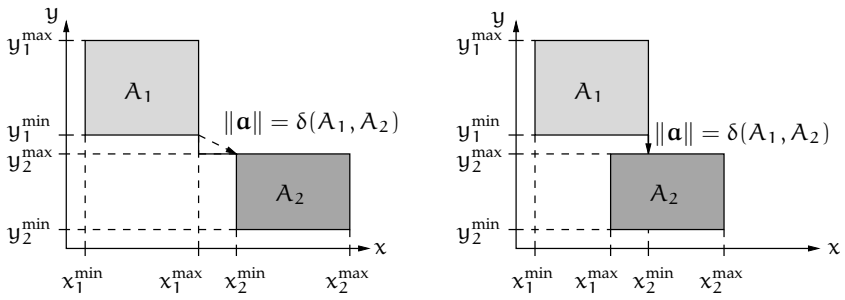
$$\begin{aligned} \text{mit} \quad & x_i^{min} = d_{i1}^{min}\,, & & x_i^{max} = d_{i1}^{max}\,, \\ & y_i^{min} = d_{i2}^{min}\,, & & y_i^{max} = d_{i2}^{max}\,, \\ & z_i^{min} = d_{i3}^{min}\,, & & z_i^{max} = d_{i3}^{max}\,. \end{aligned}$$

Abbildung 3.31 veranschaulicht diese Überlegungen für den zweidimensionalen Fall.

Wir betrachten nun o.B.d.A. die x-Koordinatenachse. Falls die Intervalle X_1 und X_2 sich nicht überlappen, so bestimmen die am nahesten zueinander gelegenen Intervallgrenzen die x-Komponente des Abstandsvektors $\mathbf{a}$. Andernfalls unterscheiden sich die nahesten Punkte zwischen A_1 und A_2 bezüglich der x-Koordinate nicht, so daß wir die entsprechende Komponente von $\mathbf{a}$ auf Null setzen können. Wir erhalten somit für die x-Komponente des von A_1 zu A_2 gerichteten Abstandsvektors $\mathbf{a}$:

$$a_1 = \begin{cases} x_2^{max} - x_1^{min} & \text{falls } x_2^{max} < x_1^{min}\,; \\ x_2^{min} - x_1^{max} & \text{falls } x_2^{min} > x_1^{max}\,; \\ 0 & \text{sonst}\,. \end{cases}$$

Abbildung 3.31 Der Euklidische Abstand zwischen zwei achsenorientierten Boxen.



Die y- bzw. z-Komponente ergibt sich analog.
Da wir eine achsenorientierte Box A_i durch die Vektoren $\mathbf{d}_i^{min}$ und $\mathbf{d}_i^{max}$ beschreiben, erhalten wir für $1 \leq j \leq 3$:

$$a_j = \begin{cases} d_{2j}^{max} - d_{1j}^{min} & \text{falls } d_{2j}^{max} < d_{1j}^{min}\,; \\ d_{2j}^{min} - d_{1j}^{max} & \text{falls } d_{2j}^{min} > d_{1j}^{max}\,; \\ 0 & \text{sonst}\,. \end{cases}$$

Der quadratische Euklidische Abstand zwischen zwei achsenorientierten Boxen $A_i = AABB(\mathbf{d}_i^{min}, \mathbf{d}_i^{max})$, $i = 1, 2$ ergibt sich als:

$$\delta^2(A_1, A_2) = \|\mathbf{a}\|^2 = \sum_{j=1}^{3} a_j^2\,.$$

Des weiteren betragen die Kosten der Abstandsberechnung zwischen den beiden achsenorientierten Boxen:

$$C_D(A_1, A_2) = 3\,MULT + 5\,ADD + 6\,COMP\,.$$

3.8 Simultane Hierarchietraversierung

Wir wollen in diesem Abschnitt die Ausgestaltung des Branch-and-Bound-Verfahrens zur Abstandsberechnung abschließen, indem wir die in diesem Kapitel vorgestellten Datenstrukturen und Algorithmen in eine sinnvolle *Ablaufsteuerung* integrieren.

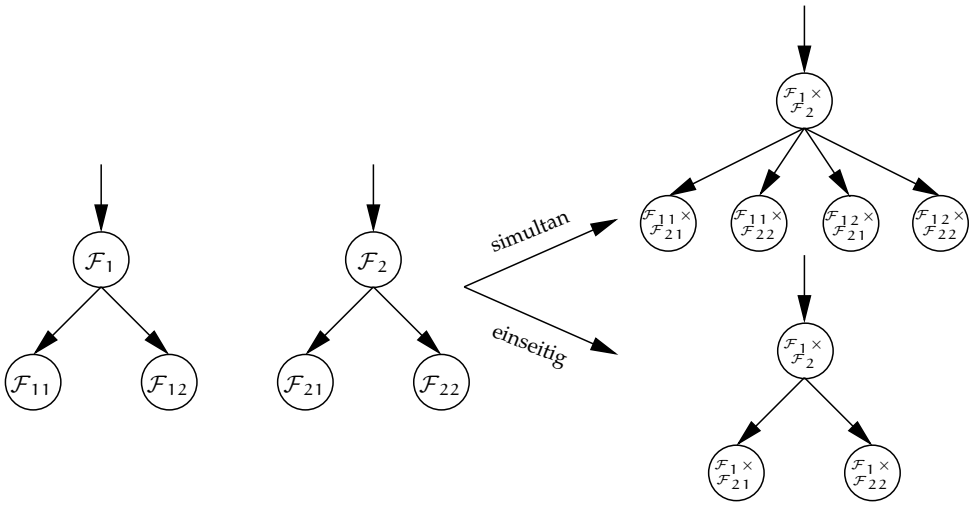
Die Datenstruktur, die dem Verfahren zugrundeliegt, ist der Hüllkörperbaum, der die Flächenmenge des Körpers von Hierarchiestufe zu Hierarchiestufe zunehmend feiner partitioniert und die so entstehenden Teilmengen durch Hüllkörper überdeckt. Beim Abstandstest zweier Körper sind wir gezwungen, die Menge der Flächenpaare $X = \mathcal{F}_1 \times \mathcal{F}_2$ auf ihr Abstandsminimum hin zu untersuchen. Das Branch-and-Bound-Verfahren zerlegt diesen Suchraum in Subräume, mit dem Ziel, möglichst viele Teilprobleme als Lieferant des Abstandsminimums ausschließen zu können. Der Ausschlußtest für einen solchen Subraum basiert dabei auf dem Vergleich des Hüllkörperabstands der beiden den Subraum definierenden Flächenmengen mit einer oberen Schranke für den Abstand der beiden Körper. Obere Schranken sind einfach zu finden, da jeder Abstand zwischen einem Paar von Oberflächenelementen der beiden Körper einen solchen Wert definiert. Gelingt es dem Verfahren nicht, ein Teilproblem auszuschließen, da der Hüllkörperabstand kleiner als die obere Schranke ist, so wird das Problem weiter zerlegt, d.h. die betrachtete Menge von Flächenpaaren weiter partitioniert.

3.8.1 Der Vergleichsbaum als Ergebnis der Expansionsstrategie

Die gerade beschriebene Vorgehensweise können wir als Vergleich der Hüllkörperhierarchien der beiden Körper $\mathcal{K}_1$ und $\mathcal{K}_2$ auffassen. Man betrachtet zunächst die Wurzeln der entsprechenden D-Bäume $D(\mathcal{K}_i)$, $i = 1, 2$. Diese repräsentieren die Menge aller Flächen der beiden Körper, nämlich $\mathcal{F}_1$ und $\mathcal{F}_2$. Sofern das eingeschlossene Objekt und sein Hüllkörper nicht identisch sind, wird der Hüllkörperabstand kleiner als die obere Schranke sein, so daß wir einen oder beide Wurzelknoten expandieren müssen. Auf diese Weise ergeben sich weitere zu betrachtende Knotenpaare, wobei die durch sie definierten Paare von Flächenmengen eine Partitionierung des Ausgangsproblems darstellen.

Es ist daher äußerst hilfreich, die Traversierung der beiden Bäume als Durchmusterung eines einzigen imaginären *Vergleichsbaumes* aufzufassen. Jeder Knoten v dieses Baumes entspricht dabei einem Paar von Knoten (v_1, v_2) der beiden D-Bäume und somit einer Teilmenge $\mathcal{F} = \mathcal{F}_{v_1} \times \mathcal{F}_{v_2}$ von X, deren Abstandsminimum man bestimmen möchte. Die Kinder eines Knotens des Vergleichsbaumes werden durch die *Expansionsstrategie* festgelegt. Falls beide Knoten v_1 und v_2 gleichzeitig expandiert werden (*simultane Expansion*), erhält man im Vergleichsbaum vier Kinderknoten v_{ij}, $1 \leq i, j \leq 2$, die den Kinderknotenpaaren der beiden expandierten Hüllkörperhierarchieknoten entsprechen (Abbildung 3.32). Folgt man dagegen der Strategie der *einseitigen Expansion*, so wird lediglich einer der beiden Hüllkörperhierarchieknoten durch seine Kinderknoten ersetzt. Im Vergleichsbaum ergeben sich somit nur zwei Kinderknoten. Selbstverständlich können auch beide Expansionsalternativen im Rahmen einer *hybriden Vorgehensweise* [Eck98] auf den Vergleichsbaum angewendet werden. Sobald man in einem

Abbildung 3.32 Die Strategie der *simultanen* und *einseitigen* Expansion.



der beiden Hüllkörperhierarchien ein Blatt erreicht, ist man gezwungen, eine einseitige Expansion des gerade besuchten Knotens des Vergleichsbaums durchzuführen. Diese Konstruktion wird nun solange wiederholt, bis man auch im anderen Hüllkörperbaum ebenfalls an einem Blatt angekommen ist. Die beiden Blätter der D-Bäume definieren nun ein Blatt des Vergleichsbaumes. Somit entsprechen die Blätter des Vergleichsbaumes unabhängig von der Expansionsstrategie den Paaren von Blättern der beiden Hüllkörperbäume.

Der Vergleichsbaum bleibt jedoch eine gedankliche Konstruktion, die durch die Durchmusterung der beiden Hierarchiebäume während der Laufzeit beschrieben wird. Sowohl in der Literatur als auch in konkreten Anwendungen dominiert die Strategie der simultanen Expansion. Diese erweist sich immer dann als günstig, wenn die Verbesserung der Approximationsgüte bei dem Übergang zur nächsten Hierarchiestufe für beide Bäume annähernd gleich ist. Die einseitige Expansion eines Knotenpaares ist dann vorzuziehen, wenn die Approximationsgüte der Kinderknoten in einem Hüllkörperbaum deutlich besser ist als in dem anderen. Dieser Fall tritt beispielsweise dann ein, wenn die Objekte sehr unterschiedliche Größe besitzen. Die Zahl der Hüllkörpertests ist nun geringer, wenn zunächst für das größere Objekt die Regionen ausgeschlossen werden, die zur Bestimmung des minimalen Abstands nicht in Frage kommen. Die hybride Expansionsstrategie, wie sie von ECKSTEIN [Eck98] für den Kollisionserkennungstest vorgeschlagen wird, versucht die beiden anderen Verfahren zusammenzuführen, indem an dem gerade betrachteten Knotenpaar die Expansion durchgeführt wird, die eine vorgegebene Zielgröße optimiert. In dieser Zielgröße muß dabei sowohl die Veränderung der Approximationsgüte als auch der Aufwand zusätzlicher Hüllkörpertests berücksichtigt werden.

Im folgenden werden wir verschiedene Verfahren vorstellen, die den Ablauf der

Durchmusterung des Vergleichsbaums steuern. Zunächst werden wir Algorithmen betrachten, die ausschließlich lokale Verzweigungsentscheidungen treffen. Dabei handelt es sich um das *indifferente Verfahren* und das *lokale Greedy-Verfahren*. Aufbauend auf letzterem haben wir einen Algorithmus, den sogenannten *globalen Greedy-Algorithmus* entwickelt, der alle bisher betrachteten Vergleichsknoten bei der Auswahl eines günstigen Kinderknotens zur Fortsetzung der Durchmusterung berücksichtigt.

3.8.2 Das indifferente Verfahren

Wie bereits erwähnt, trifft dieses Verfahren ausschließlich lokale Verzweigungsentscheidungen. Es muß also für den gerade besuchten Knoten des Vergleichsbaums entscheiden, an welchem Kinderknoten die Traversierung fortgesetzt werden soll, oder in anderen Worten, welches Teilproblem als erstes gelöst werden soll. Der Vorgehensweise des indifferenten Verfahrens liegt die Annahme zugrunde, daß alle vier Teilprobleme gleichberechtigt sind, so daß die Kinder des Vergleichsbaumes in einer beliebigen Reihenfolge besucht werden können. Diese Unterstellung liefert eine einfache Umsetzung des Branch-and-Bound-Verfahrens, die in Algorithmus 19 dargestellt ist.

Die Fallunterscheidung bezüglich des Knotens $\nu = (\nu_1, \nu_2)$ hat zum Ziel, die Zahl der zu besuchenden Kinderknoten zu bestimmen. Betrachten wir zunächst den Fall, daß ν ein Blatt des Vergleichsbaumes ist, d.h. daß weder ν_1 noch ν_2 Kinder innerhalb ihrer D-Bäume besitzen. Die Variable d_u entspricht der globalen oberen Schranke, die wir immer dann zu aktualisieren versuchen, wenn wir ein solches Paar von Blättern der Hüllkörperhierarchien erreichen (Zeile 3). Der Abstand der Flächenmengen, die durch die Blätter repräsentiert werden, ersetzt die bisherige obere Schranke, falls die berechnete Distanz deren aktuellen Wert unterschreitet. Falls wir auf einen inneren Knoten des Vergleichbaumes treffen, ist eine Verzweigungsentscheidung zu treffen. Nachdem wir die Hüllkörper bezüglich der Objektbewegung aktualisiert haben, ermitteln wir für jedes Kinderknotenpaar den Abstand der Hüllkörper und vergleichen diese untere Schranke mit dem minimalen bisher beobachteten Abstand zwischen Oberflächenelementen der beiden Körper. Zeigt die untere Schranke, daß wir in dem durch den Kinderknoten definierten Teilbaum keinen besseren Abstand als d_u finden können, so verzichten wir auf eine Verzweigung in diesen Ast des Vergleichsbaumes. Andernfalls starten wir einen rekursiven Aufruf auf dem Teilproblem, d.h. wir steigen eine Hierarchieebene ab.

Die Notation von Algorithmus 19 verzichtet auf die Darstellung der durchzuführenden Hüllkörperaktualisierungen. Da diese Operationen jedoch entscheidenden Einfluß auf die Laufzeit des Verfahrens haben können, müssen wir ihren Einsatz wohl überlegen. Um den Algorithmus präziser formulieren zu können, führen wir daher die Prozedur ν_i.UPDATE ein, die den Hüllkörper $H(\nu_i)$ bezüglich der Bewegungstransformation τ aktualisiert. Daneben verwenden wir die Funktion LOWBOUND, die den Abstand der mit ν_i bzw. ν_j assoziierten Hüllkörperpaare berechnet. Die Funktion UPBOUND führt einen elementaren Abstandstest zwischen den beiden Flächenmengen durch, die durch die Blätter ν_i und ν_j repräsentiert werden.

In 3.7.2 haben wir gesehen, daß es genügt, stets einen der beiden Hüllkörper zu

Algorithmus 19 Ein Branch-and-Bound-Algorithmus mit indifferenter Verzweigungsstrategie zur Bestimmung des Abstands zweier Flächenmengen,

Eingabe: Zwei Knoten v_1, v_2 der D-Bäume von Körper 1 bzw. Körper 2.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
NODEDISTANCE(v1, v2)
(1)   if v1 is a leaf
(2)     if v2 is a leaf
(3)       du ← min{du, δ(F1, F2)}
(4)     else
(5)       foreach child p2 of v2
(6)         dl ← δ(H(v1), H(p2))
(7)         if dl < du
(8)           NODEDISTANCE(v1, p2)
(9)   else
(10)    if v2 is a leaf
(11)      foreach child p1 of v1
(12)        dl ← δ(H(p1), H(v2))
(13)        if dl < du
(14)          NODEDISTANCE(p1, v2)
(15)    else
(16)      foreach child p1 of v1
(17)        foreach child p2 of v2
(18)          dl ← δ(H(p1), H(p2))
(19)          if dl < du
(20)            NODEDISTANCE(p1, p2)
(21)  return du
```

transformieren, um deren Abstand korrekt berechnen zu können. Diese Beobachtung ist in zwei Fällen hilfreich. Betrachten wir einen inneren Knoten des Vergleichsbaums, so sind lediglich die Hüllkörper der Kinder *eines* Baumes zu aktualisieren. Noch besser stellt sich die Situation dar, wenn wir bei der Traversierung des Vergleichsbaums auf ein Knotenpaar treffen, bei dem genau einer der beiden Knoten ein Blatt des Hüllkörperbaumes darstellt (Zeile 2 bzw. 4). Der Algorithmus verbleibt in dem Blattknoten, bis er durch Rekursion ein Blatt des andern Baumes erreicht hat. Der davon betroffene Rekursionszweig kann von nicht unbedeutender Länge sein, so daß eine einmalige Transformation des Blattknotens auch alle im Rahmen des Rekursionsabstiegs besuchten Knotenpaare aktualisiert. Wir benutzen daher die Prozedur LEAFDISTANCE, die durch Algorithmus 20 beschrieben wird, um den Rekursionszweig effizient, d.h. ohne weitere Transformationen abarbeiten zu können.

Der ursprüngliche NODEDISTANCE-Algorithmus muß wie in Algorithmus 21 dar-

Algorithmus 20 Rekursionszweig des NODEDISTANCE-Algorithmus im Fall von Blattknoten der Hüllkörperhierarchien.

Eingabe: Zwei Knoten v_1, v_2, von denen mindestens einer ein Blatt seines D-Baumes darstellt.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
LEAFDISTANCE(v1, v2)
(1)   if v1 is a leaf
(2)     if v2 is a leaf
(3)       du ← min{du, UPBOUND(v1, v2)}
(4)     else
(5)       foreach child p2 of v2
(6)         dl ← LOWBOUND(v1, p2)
(7)         if dl < du
(8)           LEAFDISTANCE(v1, p2)
(9)   else
(10)    foreach child p1 of v1
(11)      dl ← LOWBOUND(p1, v2)
(12)      if dl < du
(13)        LEAFDISTANCE(p1, v2)
(14)  return du
```

gestellt modifiziert werden.

3.8.3 Das lokale Greedy-Verfahren

Bisher haben wir Informationen über die Abstände der Hüllkörper lediglich zur Formulierung einer Ausschlußregel für Teilbäume genutzt. Es ist jedoch naheliegend, solche untere Schranken als heuristisches Entscheidungskriterium für die Festlegung einer Reihenfolge einzusetzen, in der die Kinderknotenpaare, d.h. die Teilbäume betrachtet werden sollen.

Dem hier vorgestellten *Greedy-Verfahren* liegt die Annahme zugrunde, daß das Knotenpaar, das den minimalen Hüllkörperabstand aufweist, dem Teilbaum entspricht, in dem die Wahrscheinlickeit, das naheste Punktepaar zu finden, am größten ist.

Bei der *lokal agierenden* Greedy-Heuristik, wählen wir daher im Rahmen der Verzweigungsentscheidung an einem Knoten des Vergleichsbaums den Kinderknoten mit kleinstem Hüllkörperabstand. Der entsprechende Pseudoalgorithmus benutzt eine neue Funktion MININDEX, die das zweidimensionale Feld der Hüllkörperabstände, d_l, nach dem Kinderknoten des Vergleichsbaumes (v_i, v_j) mit minimalem Abstand durchsucht und dessen Indizes (i, j) zurückgibt. Des weiteren verwenden wir eine Routine v_i.CHILD(j), die das j-te Kind des Knotens v_i liefert.

Algorithmus 21 Eine Variante des NODEDISTANCE-Algorithmus mit minimaler Anzahl an Aktualisierungsoperationen.

Eingabe: Zwei Knoten v_1, v_2 der D-Bäume von Körper 1 bzw. Körper 2.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
NODEDISTANCE(v1, v2)
(1)   if v1 is a leaf
(2)     return LEAFDISTANCE(v1, v2)
(3)   else if v2 is a leaf
(4)     v2.UPDATE(τ^-1)
(5)     return LEAFDISTANCE(v1, v2)
(6)   else
(7)     foreach child p1 of v1
(8)       p1.UPDATE(τ)
(9)       foreach child p2 of v2
(10)        d_l ← LOWBOUND(p1, p2)
(11)        if d_l < d_u
(12)          NODEDISTANCE(p1, p2)
(13)        return d_u
```

Algorithmus 22 beschreibt diese Variante des NODEDISTANCE-Verfahrens. Die lokale Greedy-Strategie wird selbstverständlich auch in der Rountine LEAFDISTANCE angewendet, deren Modifikation durch Algorithmus 23 beschrieben wird.

3.8.4 Das globale Greedy-Verfahren

Die Greedy-Strategie, die wir in Algorithmus 22 lediglich lokal an einem Knoten des Vergleichsbaumes angewendet haben, kann durch den Einsatz komplexerer Datenstrukturen auf die Menge der bisher betrachteten Knoten ausgeweitet werden. Dabei wollen wir unser Greedy-Entscheidungskriterium nicht nur lokal an dem gerade besuchten Knoten anwenden, sondern den nächsten Knoten der Vergleichsbaumtraversierung unter allen bisher besuchten, aber noch nicht expandierten Kinderknoten wählen. Die Umsetzung dieser Strategie erfordert eine Datenstruktur, die eine sortierte Folge der noch nicht expandierten Knoten des Vergleichsbaumes aufrechterhält. Jeder besuchte Knoten, der kein Blatt des Vergleichsbaumes darstellt, wird unter dem Schlüssel seines Hüllkörperabstands in diese Folge eingefügt. Treffen wir dagegen auf ein Blatt, so liefert dieses analog zu unserer bisherigen Vorgehensweise, eine obere Schranke für den Abstand der beiden Körper. Daher können wir alle Einträge der Sequenz, deren Schlüssel größer oder gleich dieser oberen Schranke sind, aus der Folge löschen. Aufgrund der Sortierung der Sequenz, entsprechen diese Löschoperationen einer Kürzung der Folge ausgehend von dem Eintrag mit größtem Schlüssel. Die Verzweigungsentscheidung wird durch Expansion des Knotens mit kleinstem Schlüssel beantwortet.

Algorithmus 22 Variante des NODEDISTANCE-Algorithmus unter Einsatz der lokalen Greedy-Strategie.

Eingabe: Zwei Knoten v_1, v_2 der D-Bäume von Körper 1 bzw. Körper 2.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
NODEDISTANCELG(v1, v2)
(1)    if v1 is a leaf
(2)       return LEAFDISTANCELG(v1, v2)
(3)    else if v2 is a leaf
(4)       v2.UPDATE(τ^-1)
(5)       return LEAFDISTANCELG(v1, v2)
(6)    else
(7)       for i ← 1 to 2
(8)          p1 ← v1.CHILD(i)
(9)          p1.UPDATE(τ)
(10)         for j ← 1 to 2
(11)            dl[i][j] ← LOWBOUND(p1, v2.CHILD(j))
(12)      for k ← 1 to 4
(13)         (i, j) ← MININDEX(dl)
(14)         if dl[i][j] ≥ du
(15)            return du
(16)         dl[i][j] ← ∞
(17)         NODEDISTANCELG(v1.CHILD(i), v2.CHILD(j))
(18)      return du
```

Dazu wird der Knoten aus der Sequenz gelöscht und seine Kinderknoten in die Folge eingestellt, falls deren Hüllkörperabstände die aktuelle obere Schranke unterbieten.

Die gesuchte Datenstruktur muß also folgende Operationen bereitstellen:

1. MINKEY(): Liefert den kleinsten Schlüssel.
2. MININFO(): Liefert den Knoten mit minimalem Schlüssel.
3. MAXKEY(): Liefert den größten Schlüssel.
4. INSERT(k, (v_1, v_2)): fügt den Knoten mit Hüllkörperabstand k in die Sequenz ein.
5. DELTEMIN(): Entfernt den Eintrag mit minimalem Schlüssel.
6. DELETEMAX(): Entfernt den Eintrag mit maximalem Schlüssel.

Eine einfache Implementierung dieser Datenstruktur benutzt eine doppelt verkettete Liste. Dabei können wir hoffen, daß die Schlüssel der Kinderknoten sich nicht allzu sehr von denen des expandierten Knotens unterscheiden, so daß INSERTIONSORT ein effizientes Sortierverfahren für die nahezu geordnete Liste darstellt. Diese Annahme ist bei Expansion auf den ersten Hierarchieebenen nur selten gerechtfertigt, da hier im

Algorithmus 23 Variante des LEAFDISTANCE-Algorithmus unter Einsatz der lokalen Greedy-Strategie.

Eingabe: Zwei Knoten v_1, v_2, von denen mindestens einer ein Blatt seines D-Baumes darstellt.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
LEAFDISTANCELG(v1, v2)
(1)   if v1 is a leaf
(2)     if v2 is a leaf
(3)       return du ← min{du, UPBOUND(v1, v2)}
(4)     else
(5)       dl[1] ← LOWBOUND(v1, v2.CHILD(1))
(6)       dl[2] ← LOWBOUND(v1, v2.CHILD(2))
(7)       if dl[1] < dl[2]
(8)         if dl[1] < du then LEAFDISTANCELG(v1, v2.CHILD(1))
(9)                       else return du
(10)        if dl[2] < du then LEAFDISTANCELG(v1, v2.CHILD(2))
(11)                      else return du
(12)      else
(13)        if dl[2] < du then LEAFDISTANCELG(v1, v2.CHILD(2))
(14)                      else return du
(15)        if dl[1] < du then LEAFDISTANCELG(v1, v2.CHILD(1))
(16)                      else return du
(17)  else
(18)    dl[1] ← LOWBOUND(v2, v1.CHILD(1))
(19)    dl[2] ← LOWBOUND(v2, v1.CHILD(2))
(20)    if dl[1] < dl[2]
(21)      if dl[1] < du then LEAFDISTANCELG(v2, v1.CHILD(1))
(22)                    else return du
(23)      if dl[2] < du then LEAFDISTANCELG(v2, v1.CHILD(2))
(24)                    else return du
(25)    else
(26)      if dl[2] < du then LEAFDISTANCELG(v2, v1.CHILD(2))
(27)                    else return du
(28)      if dl[1] < du then LEAFDISTANCELG(v2, v1.CHILD(1))
(29)                    else return du
```

allgemeinen eine deutliche Verbesserung der Approximationsgüte erzielt wird. Gleichzeitig ist jedoch die Zahl der Sequenzeinträge gering, so daß die Sortierkosten beim Einfügen noch nicht ins Gewicht fallen. Die INSERT-Operationen werden erst dann kritisch, wenn die Liste im Rahmen der Baumdurchmusterung deutlich angewachsen

Algorithmus 24 Ein Algorithmus zum Einfügen der binären Teilbäume des Vergleichsbaumes in die sortierte Sequenz.

Eingabe: Zwei Knoten v_1, v_2, von denen genau einer ein Blatt seines D-Baumes darstellt.
Ausgabe: Die sortierte Sequenz, in die der durch (v_1, v_2) definierte Teilbaum eingefügt wurde.

```
INSERTCHILDREN(v1, v2)
(1)   if v1 is a leaf
(2)     foreach child p2 of v2
(3)       d ← LOWBOUND(v1, p2)
(4)       if d < du
(5)         S.INSERT(d, (v1, p2))
(6)         INSERTCHILDREN(v1, p2)
(7)   else
(8)     foreach child p1 of v1
(9)       d ← LOWBOUND(p1, v2)
(10)      if d < du
(11)        S.INSERT(d, (p1, v2))
(12)        INSERTCHILDREN(p1, v2)
(13)  return S
```

ist. In diesem Fall sind wir in den Hüllkörperhierarchien jedoch so tief abgestiegen, daß die Approximationsgüte und damit die Schlüssel der einzufügenden Knoten keine drastischen Verbesserungen gegenüber dem Schlüssel des Vaterknoten darstellen.

Eine aufwendigere aber auch effizientere Datenstruktur zur Realisierung einer sortierten Folge ist die sogenannte *Skiplist*. Die Software-Bibliothek LEDA [MN99] stellt eine effiziente Implementierung dieser randomisierten Datenstruktur bereit. Der Zugriff auf das kleinste bzw. größte Glied der Folge sowie das Löschen der entsprechenden Einträge ist dabei in konstanter Zeit möglich. Das Einfügen eines Knotens in die Sequenz verursacht dagegen erwartete Kosten von $O(\log n)$, wobei n die Anzahl der Folgenglieder bezeichnet. Es hat sich in unseren Tests herausgestellt, daß die Verwendung einer Skiplist die effizienteste Implementierung des Verfahrens erlaubt.

Der Algorithmus, der unser Branch-and-Bound-Verfahren aus 3.3.4 mit Hilfe der globalen Greedy-Strategie umsetzt, unterscheidet sich von den bisher vorgestellten Verfahren durch nahezu vollkommenen Verzicht auf Rekursion. Der Teil des Vergleichsbaumes, den unserer Algorithmus betrachtet, wird statt dessen mit Hilfe der Skiplist-Datenstruktur aufgezählt. Die durch sie repräsentierte sortierte Folge wird in Algorithmus 25 mit S bezeichnet. Um die Zahl der Hüllkörperaktualisierungen zu minimieren, benutzen wir die Routine INSERTCHILDREN, die durch den Pseudoalgorithmus 24 beschrieben wird. Trifft der NODEDISTANCEGG-Algorithmus auf einen Knoten (v_1, v_2) des Vergleichsbaumes, für den gilt, daß entweder v_1 oder v_2 ein Blatt des ent-

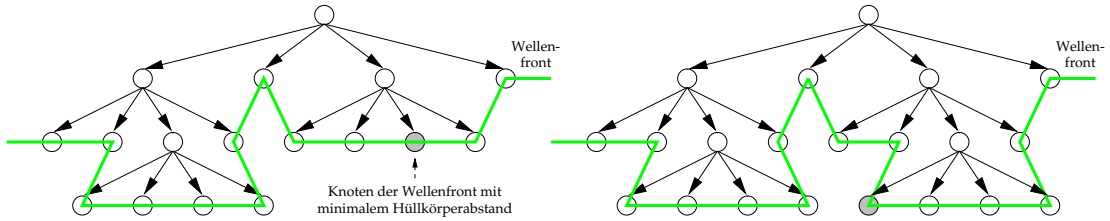
sprechenden Hüllkörperbaumes darstellt, so wird diese Routine aufgerufen, um den durch (v_1, v_2) definierten Teilbaum in die Sequenz einzustellen. Wir expandieren in diesem Fall den gesamten Teilbaum, da die einmalige Transformation des Blattknotens in (v_1, v_2) genügt, um die Hüllkörperabstände in allen Knoten des zu (v_1, v_2) gehörenden Teilbaums ohne weitere Aktualisierungsoperationen zu bestimmen. Eine Alternative zu dieser Vorgehensweise besteht darin, lediglich die beiden Kinderknoten von (v_1, v_2) in die Sequenz einzustellen, so daß man auf die Prozedur INSERTCHILDREN verzichten kann.

Algorithmus 25 Variante des NODEDISTANCE-Algorithmus unter Einsatz der globalen Greedy-Strategie.

Eingabe: Zwei Knoten v_1, v_2 der D-Bäume von Körper 1 bzw. Körper 2.
Ausgabe: Der Euklidische Abstand $\delta(\mathcal{F}_1, \mathcal{F}_2)$ der durch v_1 und v_2 repräsentierten Flächenmengen $\mathcal{F}_1$ bzw. $\mathcal{F}_2$.

```
NODEDISTANCEGG(v1, v2)
(1)   v1.UPDATE(τ)
(2)   d ← LOWBOUND(v1, v2)
(3)   if d < du
(4)      S.INSERT(d, (v1, v2))
(5)   while S ≠ ∅
(6)      (* Aktualisiere untere Abstandsschranke *)
(7)      dl ← min{du, S.MINKEY()}
(8)      (* Verzweigungsschritt *)
(9)      (v1, v2) ← MININFO()
(10)     S.DELMIN()
(11)     if v1 is a leaf
(12)        if v2 is a leaf then du ← min{du, UPBOUND(v1, v2)}
(13)                        else S ← INSERTCHILDREN(S, (v1, v2))
(14)     else
(15)        if v2 is a leaf
(16)           v2.UPDATE(τ^-1)
(17)           S ← INSERTCHILDREN(S, (v1, v2))
(18)        else
(19)           foreach child p1 of v1
(20)              v1.UPDATE(τ)
(21)              foreach child p2 of v2
(22)                 d ← LOWBOUND(p1, p2)
(23)                 if d < du
(24)                    S.INSERT(d, (p1, p2))
(25)     while S.MAXKEY() ≥ du
(26)        S.DELETEMAX()
(27)     return du
```

Abbildung 3.33 Die Ausbreitungsstrategie des globalen Greedy-Algorithmus.



Ein ganz entscheidender Vorteil dieser Vorgehensweise, besteht in der Möglichkeit, zu jedem Zeitpunkt der Baumtraversierung eine untere Schranke für den Abstand der beiden Körper angeben zu können. Um dies einzusehen, wollen wir uns den Prozeß der Baumdurchmusterung genauer ansehen. Wir betrachten zunächst eine Variante von Algorithmus 25, die auf die Kürzung der Sequenz in den Zeilen 25 bis 26 verzichtet. Zu einem bestimmten Zeitpunkt der Traversierung sei $S = \{s_1, \ldots, s_n\}$ mit $s_i = (v_1^i, v_2^i)$ die Menge der besuchten inneren Knoten, die der Algorithmus in der sortierten Folge hält. Des weiteren sei $W = \{w_1, \ldots, w_m\}$ mit $w_j = (w_1^j, w_2^j)$ die Menge der *bisher* betrachteten Blätter des Vergleichsbaumes. Diese definieren die obere Schranke d_u für den Abstand der zugrundeliegenden Körper K_1 und K_2:

$$d_u = \min_{1 \le j \le m} S\big(\mathcal{F}(w_1^j), \mathcal{F}(w_2^j)\big) \ge \min_{(f_1, f_2) \in \mathcal{F}_1 \times \mathcal{F}_2} \delta(f_1, f_2) = \delta(\mathcal{K}_1, \mathcal{K}_2)\,.$$

Die Knoten aus $S \cup W$ bilden ähnlich wie beim Algorithmus von DIJKSTRA [CLR94] zur Lösung des *single-source-shortest-path problem* eine *Wellenfront*, die sich kontinuierlich über den Vergleichsbaum ausbreitet. Abbildung 3.33 verdeutlicht den Prozeß. Jede Wellenfront definiert eine untere Schranke für den Abstand des zugrundeliegenden Körperpaares, da die Knoten der Wellenfront Paare von Flächenmengen der beiden Objekte repräsentieren, deren Vereinigung die Menge $X = \mathcal{F}_1 \times \mathcal{F}_2$ ergibt. Diese Beobachtung ist einsichtig, wenn man bedenkt, daß ein Knoten und somit ein Paar von Flächenmengen nur dann aus der Sequenz entfernt wird, wenn es sich um ein Blatt des Vergleichsbaumes handelt. Jeder innere Knoten wird dagegen durch seine Kinderknoten ersetzt, die eine feinere Partition der Flächenpaarmenge des Vaterknotens darstellen. Da das Verfahren mit der Wurzel des Vergleichsbaumes, d.h. mit $\mathcal{F}_1 \times \mathcal{F}_2$ startet, genügt es stets, die Mengen S und W zu betrachten.

$$\begin{aligned} \mathcal{F}_1 \times \mathcal{F}_2 &= \bigcup_{1 \le i \le n} \mathcal{F}(v_1^i) \times \mathcal{F}(v_2^i) \quad \cup \quad \bigcup_{1 \le j \le m} \mathcal{F}(w_1^j) \times \mathcal{F}(w_2^j) \\ &= \mathcal{F}_S^{\times} \cup \mathcal{F}_W^{\times}\,, \end{aligned}$$

mit $\mathcal{F}_S^{\times} := \bigcup_{1 \le i \le n} \mathcal{F}(v_1^i) \times \mathcal{F}(v_2^i)$ und $\mathcal{F}_W^{\times} := \bigcup_{1 \le j \le m} \mathcal{F}(w_1^j) \times \mathcal{F}(w_2^j)$. Eine untere Schranke liefert nun der minimale Abstand aller Hüllkörperpaare der Wellenfrontkno-

ten d_H:

$$d_H := \min\Big\{ \min_{1\le i\le n} \delta\big(H(v_1^i), H(v_2^i)\big),\ \min_{1\le j\le m} \delta\big(H(w_1^j), H(w_2^j)\big)\Big\}\,.$$

Es gilt nämlich:

$$\begin{aligned} d_H &\le \min\Big\{ \min_{(f_1,f_2)\in\mathcal{F}_S^\times} \delta(f_1,f_2),\ \min_{(f_1,f_2)\in\mathcal{F}_W^\times} \delta(f_1,f_2)\Big\} \\ &= \min_{(f_1,f_2)\in F_1\times F_2} \delta(f_1,f_2) = \delta(\mathcal{F}_1,\mathcal{F}_2) = \delta(\mathcal{K}_1,\mathcal{K}_2)\,. \end{aligned}$$

Für die Blattknoten innerhalb der Wellenfront wollen wir statt des Hüllkörperabstands den exakten Abstand der durch sie beschriebenen Flächenpaare in die untere Schranke eingehen lassen. Wir erhalten auf diese Weise die in Algorithmus 25 verwendete untere Abstandsschranke d_l:

$$d_l := \min\Big\{ \min_{1\le i\le n} \delta\big(H(v_1^i), H(v_2^i)\big),\ \min_{(f_1,f_2)\in\mathcal{F}_W^\times} \delta(f_1,f_2)\Big\}\,.$$

Die Definition von d_u liefert somit:

$$d_l = \min\Big\{ \min_{1\le i\le n} \delta\big(H(v_1^i), H(v_2^i)\big),\ d_u\Big\}\,.$$

Aufgrund der Beziehung $d_u \ge \min_{1\le j\le m} \delta\big(H(w_1^j), H(w_2^j)\big)$ liefert diese Modifikation von d_H eine im allgemeinen schärfere untere Schranke. Es gilt:

$$d_H \le d_l \le \delta(K_1,K_2) \le d_u\,.$$

Als nächstes wollen wir zeigen, daß das Kürzen der Sequenz S entsprechend der aktuellen oberen Schranke d_u keinen Einfluß auf den Wert von d_l hat. Sei S' die sortierte Folge, die aus S durch den Kürzungsprozeß (Zeile 25 bis 26) hervorgeht.

$$S' = \big\{(v_1,v_2)\in S \,\big|\, \delta\big(H(v_1),H(v_2)\big) < d_u\big\}\,.$$

Legt man nun S' statt S der Berechnung der unteren Schranke zugrunde, so erhält man

$$\begin{aligned} d_l' &:= \min\Big\{ \min_{(v_1,v_2)\in S'} \delta\big(H(v_1),H(v_2)\big),\ d_u\Big\} \\ &= \min\Big\{ \min_{(v_1,v_2)\in S} \delta\big(H(v_1),H(v_2)\big),\ d_u\Big\} \\ &= d_l\,, \end{aligned}$$

da $\delta\big(H(v_1),H(v_2)\big) \ge d_u$ für alle $(v_1,v_2)\in S\setminus S'$. Des weiteren hat der Kürzungsprozeß bis zum Zeitpunkt der Terminierung von Algorithmus 25 keinen Einfluß auf die Wahl des Expansionsknotens und damit auf den Ablauf des Verfahrens, wie die folgende Fallunterscheidung zeigt:

1. Fall: $\exists(v_1, v_2) \in S : \delta\big(H(v_1), H(v_2)\big) < d_u$
   Nach dem Kürzen von S ist jeder Knoten aus S, dessen Hüllkörperabstand kleiner als d_u ist, auch in S' enthalten, so daß gilt:

$$\min_{(v_1,v_2)\in S'} \delta\big(H(v_1), H(v_2)\big) = \min_{(v_1,v_2)\in S} \delta\big(H(v_1), H(v_2)\big) .$$

   Die Wahl des Verzweigungsknotens wird somit durch die Kürzung der Sequenz nicht beeinflußt.

2. Fall: $\forall(v_1, v_2) \in S : \delta\big(H(v_1), H(v_2)\big) \geq d_u$
   In diesem Fall kann die Betrachtung eines beliebigen Knotens aus S den bereits beobachteten Abstand d_u nicht unterbieten. Wir haben also den minimalen Abstand zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ gefunden. Nach dem Kürzen der Sequenz erhalten wir die Abbruchbedingung von Algorithmus 25: $S = \emptyset$ bzw. $d_l = d_u$.

Die globalere Sichtweise dieses Verfahrens auf die Problemstellung erlaubt sowohl eine untere, als auch eine obere Schranke für den Abstand des Körperpaares aufrechtzuerhalten und beide Schranken im Laufe der Traversierung des Vergleichsbaumes kontinuierlich zu verbessern. Die Fähigkeit eines Abstandberechnungsverfahrens zu jedem Zeitpunkt seiner Berechnung eine konservative Abschätzung der tatsächlichen Distanz angeben zu können, ist insbesondere im Rahmen von Echtzeitanwendungen von großer Bedeutung, wie wir in Abschnitt 3.9.3 erläutern werden.

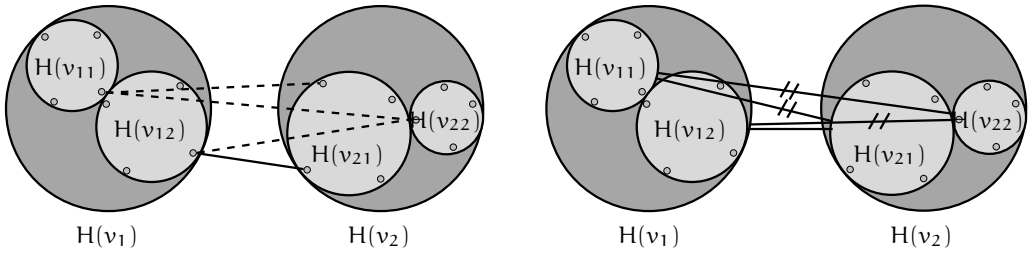
3.9 Beschleunigungsansätze

In diesem Abschnitt wollen wir einige Konzepte vorstellen, die sich in Hinblick auf die Beschleunigung des Branch-and-Bound-Verfahrens als effektiv erwiesen haben. Dabei können wir konzeptionell unterscheiden zwischen solchen Ansätzen, die eine *schnellere Konvergenz von globaler oberer Abstandsschranke und tatsächlichem Distanzwert* bewirken und Ansätzen, die eine Beschleunigung des Verfahrens durch *Approximationslösungen* erzielen. Das sogenannte *Repräsentantenkonzept* ermöglicht, die globale obere Schranke durch die Berechnung zusätzlicher lokaler oberer Schranken an den inneren Knoten des Vergleichsbaums frühzeitig zu verringern. Das *Caching*-Prinzip dagegen liefert einen guten Startwert für die obere Schranke, indem es temporäre und geometrische Kohärenz zwischen aufeinanderfolgenden Aufrufen der Abstandsberechnung ausnutzt. Anschließend werden wir darstellen, wie die *globale Greedy-Heuristik* zur Berechnung von Approximationslösungen eingesetzt werden kann. Sie unterscheidet sich von dem *universellen Fehlerschranken-Verfahren* durch die garantierte Einhaltung eines vorgegebenen Zeitbudgets.

3.9.1 Das Repräsentantenkonzept

Die Effizienz des Branch-and-Bound-Verfahrens beruht auf der Möglichkeit, Objekteile frühzeitig von einer weiteren Betrachtung auszuschließen. Das Ausschlußkriterium

Abbildung 3.34 Die Wirkungsweise des Repräsentantenkonzept.



obere Schranken durch Repräsentanten untere Schranken durch Hüllkörper

vergleicht dazu eine lokale untere Schranke für die Entfernung zweier Flächenmengen mit der globalen oberen Schranke. Diese obere Schranke wird durch die exakten Abstände von Flächenpaaren bestimmt, die wir berechnen, sobald wir im Rahmen der Baumtraversierung ein Blatt des Vergleichsbaumes erreicht haben. Dies bedeutet, daß zunächst viele Wege von der Wurzel zu den Blättern des Vergleichsbaumes durchlaufen und somit zahlreiche innere Knoten betrachtet werden müssen, bevor eine obere Schranke soweit verbessert ist, daß eine Vielzahl von Teilbäumen ausgeschlossen werden kann. Daher macht es Sinn, auch an den inneren Knoten des Baumes obere Schranken zu bestimmen, die eine frühzeitige Konvergenz von globaler oberer Schranke und tatsächlichem Distanzwert ermöglichen.

Obere Schranken sind einfach zu finden, da jedes Punktepaar der beiden Körper eine solche Approximation des tatsächlichen Körperabstands liefert. Die Idee des Repräsentantenkonzepts besteht nun darin, für jeden inneren Knoten (ν_1, ν_2) des Vergleichsbaumes eine Menge von Punktepaaren auszuwählen, die eine „gute“ obere Schranke für den Abstand der Flächenpaare $\mathcal{F}(\nu_1) \times \mathcal{F}(\nu_2)$ liefern. Dazu bestimmen wir eine Menge von Punkten der Flächenmenge $\mathcal{F}(\nu_1)$ und $\mathcal{F}(\nu_2)$, die den Knoten ν_1 bzw. ν_2 der Hüllkörperhierarchien $D(\mathcal{K}_1)$ bzw. $D(\mathcal{K}_2)$ zugrundeliegen. Diese repräsentative Punktemenge bezeichnen wir mit $R(\nu_1)$ bzw. $R(\nu_2)$. Betrachtet man die Punktepaare aus $R(\nu_1) \times R(\nu_2)$, so stellen diese mögliche Kandidaten für die Berechnung einer lokalen oberen Schranke dar.

Abbildung 3.34 verdeutlicht die Wirkungsweise des Repräsentantenkonzeptes. Die beiden großen Kreise stellen die Hüllkörper $H(\nu_1)$ und $H(\nu_2)$ des Vergleichsbaumknotens (ν_1, ν_2) dar. Darin eingeschlossen befinden dich die Hüllkörper der Kinderknoten von ν_1 und ν_2 Für jedes Kinderknotenpaar (ν_{1i}, ν_{2j}), $i, j = 1, 2$, können wir den Abstand der Repräsentanten (Punkte in/auf den Kreisen) berechnen, der eine obere Schranke für den gesamten Körperabstand darstellt. Die obere Schranke ist also nach dieser Aktualisierung höchstens so groß wie der minimale Repräsentantenabstand zwischen Knoten ν_{12} und Knoten ν_{21}. Da die Hüllkörperabstände der Kinderknotenpaare (ν_{11}, ν_{21}), (ν_{11}, ν_{22}) sowie (ν_{12}, ν_{22}) einen größeren Abstand der eingeschlossenen Flächenmenge garantieren, wird die aktuelle obere Schranke durch eine Betrachtung der

entsprechenden Teilbäume nicht verbessert werden. Einzig das Knotenpaar (v_{12}, v_{21}) kann durch Vergleich von oberer und unterer Schranke nicht ausgeschlossen werden und muß im Rahmen der Traversierung des Vergleichsbaumes betrachtet werden.

Die Auswahl der Repräsentanten können wir in der Vorberechnungsphase, d.h. beim Aufbau der Hüllkörperhierarchie durchführen. Ist ein Knoten v der Hüllkörperhierarchie erzeugt worden, so sind die drei Trägheitsachsen der Flächenmenge $\mathcal{F}(v)$, nämlich $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3\}$ bekannt. Wir erinnern uns, daß diese im Rahmen der Partitionierung von $\mathcal{F}(v)$ als Alternative für die Bestimmung einer geeigneten Schnittebenennormalen herangezogen wurden. Als Repräsentanten von $\mathcal{F}(v)$ wählen wir nun die Extrempunkte der Flächenmenge entlang der drei Trägheitsachsen:

$$R(v) \subseteq \left\{ p \in \mathcal{V}(v) \,\middle|\, \exists i\, 1 \leq i \leq 3 : \mathbf{d}_i^T \mathbf{p} = \min_{q \in \mathcal{V}(v)} \mathbf{d}_i^T \mathbf{q} \;\vee\; \mathbf{d}_i^T \mathbf{p} = \max_{\mathbf{q} \in \mathcal{V}(v)} \mathbf{d}_i^T \mathbf{q} \right\}.$$

Die Teilmengenbeziehung erklärt sich aufgrund der Tatsache, daß die Extrempunkte nicht zwangsläufig eindeutig sind, wir uns jedoch für jede Richtung d_i auf einen Punkt mit minimaler und einen mit maximaler $\mathbf{d}_i$-Projektion, $1 \leq i \leq 3$, beschränken wollen. Wir erhalten somit $|R(v)| \leq 6$ Repräsentanten der Flächenmenge $\mathcal{F}(v)$.

Im folgenden wollen wir uns der Frage zuwenden, wie die lokale obere Schranke d_R an einem inneren Knoten (v_1, v_2) des Vergleichsbaumes bestimmt werden soll. Die naheliegendste Wahl von d_R stellt das Abstandsminimum aller Punktepaare $R(v_1) \times R(v_2)$ dar. Es ist jedoch einsichtig, daß der Aufwand zur Berechnung von 36 Punktabständen in keinem Verhältnis zu den Kosten der Hüllkörperabstandsberechnung steht. Die zusätzliche Investition in den Berechnungsaufwand an inneren Knoten rechtfertigt daher nicht die erwartete schnellere Konvergenz von oberer Abstandsschranke und tatsächlichem Distanzwert. Aus diesem Grund haben wir uns für eine Vorgehensweise entschieden, die jeweils einen Punkt $p_1^* \in R(v_1)$ sowie einen Punkt $p_2^* \in R(v_2)$ bestimmt und deren Abstand als lokale obere Schranke zurückliefert. Unsere Hoffnung besteht nun darin, das Punktepaar aus $R(v_1) \times R(v_2)$ mit geringster Distanz auszuwählen. Da wir dies nicht dem Zufall überlassen wollen, verfolgen wir eine Heuristik, die zunächst die Mittelpunkte c_1, c_2 der beiden Hüllkörper $H(v_1)$ und $H(v_2)$ betrachtet. Für alle von uns vorgestellten Hüllkörpertypen haben wir einen solchen zentralen Punkt definiert. Dies erlaubt uns den von $H(v_1)$ zu $H(v_2)$ gerichteten Verbindungsvektor der Hüllkörpermittelpunkte, $\mathbf{c} = \mathbf{c}_2 - \mathbf{c}_1$ zu bestimmen. Zur Berechnung der lokalen oberen Schranke d_R wählen wir nun einen Repräsentanten aus $R(v_1)$ mit maximaler sowie einen Punkt aus $R(v_2)$ mit minimaler Projektion auf den Vektor $\mathbf{c}$:

$$d_R := \delta(p_1^*, p_2^*)$$
$$\text{mit} \quad \mathbf{c}^T \mathbf{p}_1^* = \max_{p_1 \in R(v_1)} \mathbf{c}^T \mathbf{p}_1 \qquad \mathbf{c}^T \mathbf{p}_2^* = \min_{p_2 \in R(v_2)} \mathbf{c}^T \mathbf{p}_2$$

Da sowohl die Repräsentanten als auch die Hüllkörpermittelpunkte im Rahmen des Hierarchieaufbaus bestimmt werden, müssen sie zum Zeitpunkt der Abstandsberechnung aktualisiert werden, um einen gültigen Abstandswert liefern zu können. Dazu genügt es, die entsprechenden Vektoren in das Koordinatensystem von Körper $\mathcal{K}_2$ zu

transformieren. Trotz dieses Tricks sind die Aktualisierungskosten bei sechs Repräsentanten inakzeptabel hoch, da mindestens 7 Matrix-Vektor-Multiplikationen durchgeführt werden müssen. Insgesamt wären zur Bestimmung von p_1^* und p_2^* neben diesen Matrix-Vektor-Multiplikationen 12 Skalarprodukte und 8 Vektoradditionen zu berechnen. Wir werden nun zeigen, daß die Aktualisierungsoperationen nahezu vollständig umgangen werden können. Dazu betrachten wir die folgenden äquivalenten Optimierungsprobleme:

$$\begin{array}{lll} (1) & \max \mathbf{c}^T(\mathbf{p}_1 - \mathbf{c}_1) & \min \mathbf{c}^T(\mathbf{p}_2 - \mathbf{c}_2) \quad (2) \\ & \text{s.d. } p_1 \in R(v_1) & \text{s.d. } p_2 \in R(v_2) \end{array}$$

Sei $\tau_i : \mathbf{x} \mapsto \mathbf{R}_i\mathbf{x} + \mathbf{t}_i$ die Bewegungsabbildung von Körper K_i, i=1,2, zum Zeitpunkt der Abstandsberechnung, dann ist $\tau : \mathbf{x} \mapsto \mathbf{R}\mathbf{x} + \mathbf{t}$, mit $\mathbf{R} := \mathbf{R}_2^T\mathbf{R}_1$ und $\mathbf{t} := \mathbf{R}_2^T(\mathbf{t}_1 - \mathbf{t}_2)$ die Transformation, die einen Punkt aus dem Körperkoordinatensystem von $\mathcal{K}_1$ in das System von $\mathcal{K}_2$ überführt. Unter Berücksichtigung der notwendigen Aktualisierungen entspricht der Term $\mathbf{c}^T(\mathbf{p}_1 - \mathbf{c}_1)$ dem folgenden Ausdruck:

$$\begin{aligned} & \left[\mathbf{R}_2\mathbf{c}_2 + \mathbf{t}_2 - (\mathbf{R}_1\mathbf{c}_1 + \mathbf{t}_1)\right]^T\left[\mathbf{R}_1\mathbf{p}_1 + \mathbf{t}_1 - (\mathbf{R}_1\mathbf{c}_1 + \mathbf{t}_1)\right] \\ = \; & \left[\mathbf{R}_2\mathbf{c}_2 - \mathbf{R}_1\mathbf{c}_1 - (\mathbf{t}_1 - \mathbf{t}_2)\right]^T\left[\mathbf{R}_1\mathbf{p}_1 - \mathbf{R}_1\mathbf{c}_1\right] \\ = \; & \left[\mathbf{c}_1 - \mathbf{R}^T\mathbf{c}_2 + \mathbf{R}_1^T(\mathbf{t}_1 - \mathbf{t}_2)\right]^T\mathbf{c}_1 - \left[\mathbf{c}_1 - \mathbf{R}^T\mathbf{c}_2 + \mathbf{R}_1^T(\mathbf{t}_1 - \mathbf{t}_2)\right]^T\mathbf{p}_1 . \end{aligned}$$

Das Optimierungsproblem (1) ist somit äquivalent zu:

$$\begin{array}{ll} \min & \left[\mathbf{c}_1 - \mathbf{R}^T\mathbf{c}_2 + \mathbf{R}_1^T(\mathbf{t}_1 - \mathbf{t}_2)\right]^T\mathbf{p}_1 \\ \text{s.d.} & p_1 \in R(v_1) \end{array}$$

Auch das zweite Optimierungsproblem kann auf analoge Weise vereinfacht und äquivalent formuliert werden:

$$\begin{array}{ll} \max & \left[\mathbf{c}_2 - \mathbf{R}\mathbf{c}_1 + \mathbf{R}_2^T(\mathbf{t}_1 - \mathbf{t}_2)\right]^T\mathbf{p}_2 \\ \text{s.d.} & p_2 \in R(v_2) \end{array}$$

Da die Ausdrücke $\mathbf{R}_i^T(\mathbf{t}_1 - \mathbf{t}_2)$, $i = 1, 2$, unabhängig von dem gerade besuchten Knoten sind, können diese Werte für die gesamte Baumtraversierung vorberechnet werden. Zur Bestimmung von d_R sind somit nur noch 2 Matrix-Vektor-Multiplikationen, 12 Skalarprodukte sowie 4 Vektoradditionen auszuwerten.

3.9.2 Caching nahester Punkte

Im Rahmen der Dynamiksimulation, in der die Kollisionserkennung für die gesamte Bewegungsbahn des Körpers durchgeführt werden muß, kann zeitliche und geometrische Kohärenz zwischen den einzelnen Bewegungsschritten ausgenutzt werden. Ein Zwischenspeichern (*Caching*) von Informationen, die die Kollisionserkennung im nächsten Zeitschritt beschleunigen, erlaubt eine amortisierte Kostenbetrachtung über

die Dauer der Simulation hinweg. Diese Vorgehensweise hat sich insbesondere für den Spezialfall konvexer Polyeder bewährt (vgl. Abschnitt 2.6.4). Es stellt sich daher die Frage, wie diese Idee zur Beschleunigung unseres Branch-and-Bound-Verfahrens eingesetzt werden kann. Unter der Annahme, daß die relative Lage zweier Körper sich in unmittelbar aufeinanderfolgenden Zeitschritten der Simulation nur geringfügig ändert, können wir davon ausgehen, daß das naheste Punktepaar aus dem vorangegangenen Bewegungsschritt sehr nahe am aktuellen Abstandsminimum liegt. Berechnen wir deren Distanz vor dem nächsten Aufruf des Branch-and-Bound-Verfahrens und initialisieren die globale obere Schranke mit diesem Wert, so ist zu erhoffen, daß dieser bereits so dicht am aktuellen Abstandswert liegt, daß frühzeitig Objektteile als Lieferant des nahesten Punktepaares ausgeschlossen werden können.

Die dem Konzept zugrunde liegende Annahme der *zeitlichen/geometrischen Kohärenz*, erscheint insbesondere in den folgenden Fällen gerechtfertigt:

- Falls die relative Bewegung des Körperpaares sehr gering ist und insbesondere dann, wenn beide Körper unbewegt sind.
- Falls die beiden Körper einander sehr nahe kommen und der Zeitschritt zwischen aufeinanderfolgenden Aufrufen der Abstandsberechnung klein wird.

Die Caching-Technik ist somit immer dann effektiv, wenn der Simulationsschritt keine große Änderung des kinematischen Systems bewirkt. Sie kann daher als wichtiger Schritt in Richtung der Anpassung des Berechnungsaufwands an den Berechnungsbedarf angesehen werden.

3.9.3 Approximative Abstandsinformation

In vielen Anwendungen von Dynamiksimulation spielt die Interaktion mit dem Benutzer eine entscheidende Rolle. Diese Tatsache impliziert Restriktionen, die die erlaubte Dauer der Berechnungen betreffen. Im allgemeinen handelt es sich dabei um *Echtzeitanforderungen*, denen die einzelnen Komponenten des Simulationssystems gerecht werden müssen. Laufzeitanalysen zeigen, daß nahezu immer die Kollisionserkennung den Flaschenhals innerhalb des Gesamtsystems bildet. Es ist daher wünschenswert, den Rechenzeitbedarf des Kollisionserkennungssystems über bewußte Einbußen an Ergebnisqualität steuern zu können. Im allgemeinen kann man selbstverständlich nicht erwarten, die gleichen Informationen mit geringerem Berechnungsaufwand erzielen zu können. Von außerordentlichem Interesse sind daher approximative Verfahren, die in der Lage sind, den *Trade-Off zwischen Laufzeit und Qualität* der berechneten Information wiedergeben zu können.

Im Rahmen eines solchen Approximationskonzepts muß natürlich sichergestellt sein, daß die Anforderungen anderer Komponenten, deren Berechnungen auf den Abstandsinformationen aufbauen, auch von den approximierten Distanzwerten erfüllt werden. Wir betrachten daher zunächst die Anforderungen unseres Kollisionserkennungskonzepts an die Abstandsberechnung. Offensichtlich ist nicht jeder Abstandswert, der von der exakten Distanz abweicht, zur Berechnung unterer Kollisionszeitschranken verwertbar. Lediglich konservative Schranken für den minimalen Abstand

der Körper stellen sicher, daß die Verfahren zur Ermittlung des frühsten Kollisionszeitpunktes tatsächlich untere Schranken für diesen liefern. Jeder andere Abstandswert führt dagegen zu einer Kollisionszeitabschätzung, die eine kollisionsfreie Bewegung der beiden Körper nicht mehr garantieren kann. Wir werden im folgenden zwei Verfahren vorstellen, die eine untere und somit zulässige Abstandsschranke berechnen.

Die globale Greedy-Heuristik

Das triviale approximative Abstandsberechnungsverfahren, das bei Eintritt eines vorgegebenen Abbruchkriteriums die Baumtraversierung beendet und die bis zu diesem Zeitpunkt gesammelten Abstandsinformationen zurückliefert, erfüllt nur im Fall der globalen Greedy-Heuristik aus Abschnitt 3.8.4 die Anforderungen des Kollisionserkennungskonzepts. Im selben Abschnitt haben wir gesehen, daß mit Hilfe einer sortierten Folge, in der die zu expandierenden Vergleichsbaumknoten entsprechend ihres Hüllkörperabstands gespeichert werden, eine untere Schranke für den Abstand der beiden Körper zu jedem Zeitpunkt der Baumdurchmusterung angegeben werden kann. Alle anderen vorgestellten Traversierungsverfahren sind lediglich in der Lage, die aktuelle obere Schranke als approximative Abstandsinformation zurückzuliefern. Die Fähigkeit der globalen Greedy-Heuristik, eine untere Abstandsschranke während der Baumdurchmusterung aufrechterhalten und kontinuierlich verbessern zu können, rechtfertigt den Einsatz einer komplexen Datenstruktur, wie Skiplist.

Mit der berechneten konservativen Distanzabschätzung liefert sie nicht nur zulässige Abstandsinformationen an das Kollisionserkennungssystem, sondern erlaubt zusätzlich die Formulierung plausibler *Abbruchskriterien*. Dabei unterscheiden wir *kosten-* und *güteinduzierte Kriterien*.

Das kosteninduzierte Verfahren

In Abschnitt 3.6.3 haben wir die Laufzeit der Abstandsberechnung durch die folgende Funktion beschrieben:

$$C = n_D C_D + n_B C_B + n_A C_A \, .$$

Analysen des Laufzeitverhaltens von Hüllkörperabstandsberechnungen sowie Aktualisierungsverfahren erlauben, die Koeffizienten C_D und C_A für die einzelnen Hüllkörpertypen zu bestimmen. Die Kosten C_B eines Elementartests sind dagegen unabhängig von der Wahl des Hüllkörpertyps. Somit können wir C_D und C_A relativ zu C_B ausdrücken: $c_D := \frac{C_D}{C_B}$ und $c_A := \frac{C_A}{C_B}$. Gibt man einen Schwellenwert c_s abhängig von der zur Verfügung stehenden Rechenzeit vor, so lautet das kosteninduzierte Abbruchskriterium:

$$n_D c_D + n_B + c_A n_A > c_s \, .$$

Die Baumtraversierung bricht somit ab, falls das Laufzeitbudget durch die Zahl der durchgeführten Hüllkörpertests, Elementartests sowie Aktualisierungsprozesse aufgebraucht ist.

Das güteinduzierte Verfahren

Während das kosteninduzierte Kriterium sicherstellt, daß ein vorgegebenes Zeitbudget

nicht überschritten wird, soll das nun herzuleitende Abbruchkriterium die Einhaltung einer a priori festgelegten relativen Fehlerschranke α garantieren. Bezeichnen wir wie bisher die aktuelle untere Abstandsschranke mit d_l, so kann das Kriterium folgendermaßen formuliert werden:

$$\frac{\delta(\mathcal{K}_1, \mathcal{K}_2) - d_l}{\delta(\mathcal{K}_2, \mathcal{K}_2)} \leq \alpha \,.$$

Dieses Kriterium ist jedoch zunächst nicht anwendbar, da $\delta(\mathcal{K}_1, \mathcal{K}_2)$ als zu approximierende Größe während der Berechnung unbekannt ist. Wir müssen daher eine obere Schranke für den relativen Approximationsfehler von d_l bestimmen. Dazu vergleichen wir d_l mit der aktuellen oberen Abstandsschranke d_u. Aufgrund der Beziehung $d_l \leq \delta(\mathcal{K}_1, \mathcal{K}_2) \leq d_u$ gilt:

$$\frac{d_u - d_l}{d_u} \geq \frac{\delta(\mathcal{K}_1, \mathcal{K}_2) - d_l}{\delta(\mathcal{K}_1, \mathcal{K}_2)} \,. \tag{3.21}$$

Das folgende Abbruchkriterium, das vor jeder Expansion eines Vergleichsbaumknotens überprüft wird, garantiert somit die Einhaltung der Fehlerschranke α:

$$\frac{d_u - d_l}{d_u} \leq \alpha \,.$$

Es gilt nämlich aufgrund von 3.21:

$$(1 - \alpha)\delta(\mathcal{K}_1, \mathcal{K}_2) \leq d_l \leq \delta(\mathcal{K}_1, \mathcal{K}_2) \,.$$

Das universelle Fehlerschrankenverfahren

Das im vorangegangen Abschnitt beschriebene approximative Abstandsberechnungsverfahren ist in der Lage, zu jedem Zeitpunkt der Vergleichsbaumtraversierung eine untere Schranke für die Distanz der beiden Körper anzugeben. Mit Hilfe des kosteninduzierten Abbruchkriteriums kann somit sichergestellt werden, daß die Berechnung ein vorgegebenes Zeitlimit einhält und das Ergebnis eine zulässige Abstandsapproximation darstellt. Diese Abbruchtoleranz bezahlt man jedoch mit höheren Laufzeitkosten durch den Einsatz einer komplexen Datenstruktur, der Skiplist.

Verzichtet man auf die Möglichkeit, maximale Berechnungszeiten garantieren zu können, so lassen sich untere Abstandsschranken mit vorgegebener Approximationsgüte unabhängig von der Wahl der Traversierungsstrategie berechnen. Analog zum güteinduzierten Abbruchkriterium im vorangegangenen Abschnitt legen wir auch hier eine einzuhaltende relative Fehlerschranke α der Approximation zugrunde. Die Idee des Verfahrens ist nun die folgende. Sobald wir einen exakten Abstandswert zwischen konkreten Oberflächenelementen ermittelt haben, verkürzen wir diesen um den Faktor α und aktualisieren die globale obere Schranke analog zu unserer bisherigen Vorgehensweise, falls dies notwendig sein sollte. Durch den Vergleich der Hüllkörperabstände und der künstlich verringerten oberen Schranke ist das Branch-and-Bound-Verfahren in der Lage, Teilbäume früher auszuschließen, als dies bei der Berechnung der exakten Distanz möglich wäre. Der Algorithmus liefert schließlich einen Distanzwert d_u, der eine untere Abstandsschranke mit Approximationsfaktor α darstellt. Das folgende Lemma verifiziert diese nicht ganz offensichtliche Behauptung.

Lemma 3.16. *Aktualisiert man die globale, obere Schranke* d_u *des Branch-and-Bound-Verfahrens aus Abschnitt 3.3.4 mit den um den Faktor* α *verkürzten lokalen oberen Schranken, so gilt für den Wert* d_u^**, mit dem das Verfahren terminiert:*

$$(1-\alpha)\delta(\mathcal{K}_1,\mathcal{K}_2) \leq d_u^* \leq \delta(\mathcal{K}_1,\mathcal{K}_2)\,.$$

Beweis. Das Repräsentantenkonzept wollen wir an dieser Stelle nicht berücksichtigen, da es im ursprünglichen Algorithmus keine Anwendung findet, sondern vielmehr als Optimierungsalternative zu verstehen ist. Der Beweis kann jedoch leicht um diese Art von lokalen Schranken ergänzt werden. Somit erhält man eine lokale obere Schranke ausschließlich durch einen Elementartest zwischen dem Paar von Flächenmengen, das durch einen besuchten Blattknoten des Vergleichsbaumes definiert wird. Bezeichnen wir die Menge der Blattknoten (w_1, w_2), die im Rahmen der Baumdurchmusterung betrachtet werden, mit W, so gilt:

$$\begin{aligned} d_u^* &= \min_{(w_1,w_2)\in W} (1-\alpha)\delta\big(\mathcal{F}(w_1),\mathcal{F}(w_2)\big) \\ &= (1-\alpha) \min_{(w_1,w_2)\in W} \delta\big(\mathcal{F}(w_1),\mathcal{F}(w_2)\big) \qquad (3.22)\\ &\geq (1-\alpha) \min_{(f_1,f_2)\in \mathcal{F}_1\times\mathcal{F}_2} \delta(f_1,f_2) \\ &= (1-\alpha)\delta(\mathcal{F}_1,\mathcal{F}_2) = (1-\alpha)\delta(\mathcal{K}_1,\mathcal{K}_2)\,. \end{aligned}$$

Um zu zeigen, daß d_u^* eine untere Abstandsschranke darstellt, müssen wir uns zunächst bewußt sein, daß die Menge S, die die Wurzeln der ausgeschlossenen Teilbäume umfaßt, vereinigt mit der Menge W der tatsächlich besuchten Blattknoten eine Partition der Flächenpaarmenge $X = \mathcal{F}_1 \times \mathcal{F}_2$ induziert, also

$$\bigcup_{(v_1,v_2)\in S} \mathcal{F}(v_1)\times\mathcal{F}(v_2) \quad \cup \bigcup_{(w_1,w_2)\in W} \mathcal{F}(w_1)\times\mathcal{F}(w_2) = \mathcal{F}_S^\times \cup \mathcal{F}_W^\times = \mathcal{F}_1\times\mathcal{F}_2\,,$$

mit $\mathcal{F}_S^\times := \bigcup_{(v_1,v_2)\in S} \mathcal{F}(v_1)\times\mathcal{F}(v_2)$ und $\mathcal{F}_W^\times := \bigcup_{(w_1,w_2)\in W} \mathcal{F}(w_1)\times\mathcal{F}(w_2)$.
Das Branch-and-Bound-Ausschlußkriterium impliziert nun:

$$d_u^* \leq \delta\big(\mathcal{F}(v_1),\mathcal{F}(v_2)\big) \quad \forall (v_1,v_2)\in S\,.$$

Zusätzlich wissen wir aufgrund von 3.22:

$$d_u^* = (1-\alpha) \min_{(w_1,w_2)\in W} \delta(\mathcal{F}(w_1),\mathcal{F}(w_2))\,.$$

Es gilt daher:

$$\begin{aligned} d_u^* &\leq \min\Big\{ \min_{(v_1,v_2)\in S} \delta\big(\mathcal{F}(v_1),\mathcal{F}(v_2)\big), \min_{(w_1,w_2)\in W} \delta\big(\mathcal{F}(w_1),\mathcal{F}(w_2)\big)\Big\} \\ &= \min\Big\{ \min_{(f_1,f_2)\in S^\times} \delta(f_1,f_2), \min_{(f_1,f_2)\in W^\times} \delta(f_1,f_2)\Big\} \\ &= \min\big\{\delta(f_1,f_2)\,\big|\,(f_1,f_2)\in \mathcal{F}_S^\times \cup \mathcal{F}_W^\times\big\} \\ &= \min\big\{\delta(f_1,f_2)\big|\in \mathcal{F}_1\times\mathcal{F}_2\big\} = \delta(\mathcal{F}_1,\mathcal{F}_2) = \delta(\mathcal{K}_1,\mathcal{K}_2)\,, \end{aligned}$$

woraus unsere Behauptung folgt. □

Sowohl das kosteninduzierte, als auch das güteinduzierte Kriterium stellen plausible Abbruchkriterien da, deren Vor- und Nachteile sich komplementär gegenüberstehen. Das kosteninduzierte Verfahren garantiert die Einhaltung eines vorgegebenen Rechenzeitbudgets, führt jedoch zu Abstandsschranken, deren Approximationsgüten stark variieren können. An diesem Punkt setzt das güteinduzierte Kriterium an, das eine vorgegebene Approximationsgüte zwar garantieren, die Laufzeit des Abstandsberechnungsverfahrens jedoch nicht beschränken kann.

4 Das Kollisionserkennungssystem

In diesem Kapitel wollen wir das statische Abstandsberechnungsverfahren aus Kapitel 3 zu einer dynamischen Kollisionserkennung ausbauen. Eine allgemeine Vorgehensweise auf der Basis unterer Kollisionszeitschranken haben wir bereits in Abschnitt 2.6.3 vorgestellt. Die Umsetzung dieses Konzeptes soll nun im Detail beschrieben werden. Neben der zentralen Problemstellung, der Bestimmung des frühstmöglichen Kollisionszeitpunktes zweier Körper, steht die Beschreibung des Kollisionsheaps als Datenstruktur zur Ablaufsteuerung der Kollisionstests im Mittelpunkt unserer Überlegungen.

4.1 Distanzfunktion und Kollisionszeitpunkt

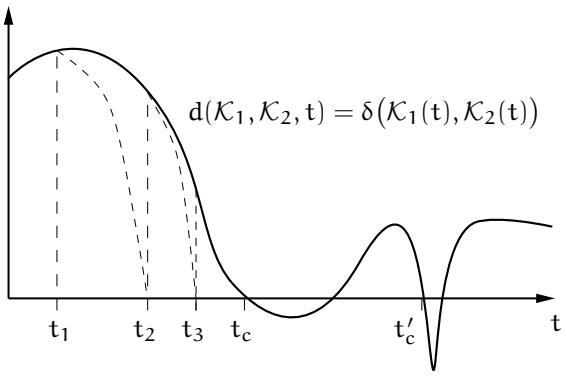
In Kapitel 3 haben wir mit dem vorgestellten Abstandsberechnungsverfahren die Grundlage des Kollisionserkennungsystems gelegt. Sofern die Bewegung zweier Körper $\mathcal{K}_1$, $\mathcal{K}_2$ innerhalb eines Zeitintervalls $[t_0, t_1]$ kollisionsfrei ist, sind wir in der Lage die *Distanzfunktion* $d(\mathcal{K}_1, \mathcal{K}_2, t) := \delta\big(\mathcal{K}_1(t), \mathcal{K}_2(t)\big)$ zu jedem Zeitpunkt $t \in [t_0, t_1]$ auszuwerten. Falls die beiden Körper innerhalb des betrachteten Zeitintervalls kollidieren, entspricht der gesuchte Kollisionszeitpunkt t_c dem frühsten Zeitpunkt innerhalb des Simulationsintervalls, an dem die Distanzfunktion den Wert 0 annimmt:

$$t_c := \inf\left\{ t \in [t_0, t_1] \,\middle|\, d(\mathcal{K}_1, \mathcal{K}_2, t) = 0 \right\}.$$

Wie wir bereits in Abschnitt 2.6.3 herausgestellt haben, gestattet die Abstandsberechnung, die Nullstelle von d *konservativ* und *mit beliebiger Genauigkeit* zu approximieren. Konservativ bedeutet, daß die Abstandsfunktion ausgehend von einer kollisionsfreien Konfiguration der beiden Körper derart abgetastet wird, daß keine Nullstelle übersehen werden kann.

Dadurch unterscheidet sich das Verfahren von den „klassischen“ Ansätzen der dynamischen Kollisionserkennung. Diese approximieren den Kollisionszeitpunkt t_c mit Hilfe von Intervallunterteilung [Bar89]. Dabei wird das Intervall $[t_0, t_1]$, das den Zeitschritt zwischen einer kollisionsfreien (t_0) und einer unfreien Konfiguration (t_1) beschreibt, mittels Interpolation der jeweiligen Körperlagen in zwei Teilintervalle $[t_0, t_1']$ und $[t_1', t_1]$ aufgespalten. Die Konfiguration zum Zeitpunkt t_1' bestimmt dabei, in welchem der beiden Teilintervalle die Nullstellensuche fortgesetzt werden soll. Der Nachteil dieser Vorgehensweise wird in Abbildung 4.1 deutlich. Die Kollision zum Zeitpunkt t_c' kann bei dieser Vorgehensweise leicht übersehen werden. Diese Aussage gilt

Abbildung 4.1 Ermittlung des frühstmöglichen t_c durch konservative Approximation der „frühsten“ Nullstelle der Distanzfunktion.



insbesondere auch für solche Verfahren, die die relative Bewegungsbahn der beiden Körper in festen Zeitschritten und auf der Basis statischer Kollisionstests abtasten.

Das von uns verwendete Verfahren umgeht dieses Problem, indem es sich bei der Nullstellensuche ausgehend von positiven Distanzwerten immer dichter an den Kollisionszeitpunkt t_c herantastet, diesen jedoch niemals überschreitet. Dazu werden solange untere Schranken $t_1, t_2, \ldots t_n$ für t_c berechnet bis die Approximation t_n ausreichend dicht an den tatsächlichen Kollisionszeitpunkt heranreicht. Die Präzision, die der Bestimmung von t_c zugrunde liegt, wird durch den Parameter $\varepsilon_c > 0$ beschrieben, so daß wir als approximativen Kollisionszeitpunkt $\hat{t}_c$ den folgenden Wert erhalten:

$$\hat{t}_c = \inf\left\{ t \in [t_0, t_1] \,\middle|\, 0 \leq d(\mathcal{K}_1, \mathcal{K}_2, t) < \varepsilon_c \right\} .$$

Abbildung 4.1 veranschaulicht die Vorgehensweise. Der entscheidenden Frage, nämlich wie solche unteren Kollisionszeitschranken berechnet werden können, wollen wir uns im folgenden zuwenden.

4.2 Untere Kollisionszeitschranken

Die Berechnung unterer Kollisionszeitschranken ist der Punkt, an dem geometrische Informationen über die Lage und den Abstand der beiden Körper mit Dynamikdaten wie Geschwindigkeit und Beschleunigung der Körper kombiniert werden müssen. Da unserer Simulation die Annahme zugrunde liegt, daß die Körper sich auf ballistischen Bahnen zwischen zwei Kollisionszeitpunkten bewegen, benötigen wir neben der geometrischen Information lediglich die aktuellen Geschwindigkeitsdaten der beiden Objekte, um die Bewegung im Intervall $[t_0, t_1]$ verfolgen zu können. Unsere Aufgabe besteht also darin, eine untere Schranke für den Zeitschritt $t \in [0, T]$ mit $T = t_2 - t_1$ zu finden, der t_0 vom tatsächlichen Kollisionszeitpunkt t_c trennt.

Dazu betrachten wir die Bahnkurve eines beliebigen Punktes x_1 bzw. x_2 von $\mathcal{K}_1$ bzw. $\mathcal{K}_2$. Diese kann folgendermaßen parametrisiert werden:

$$\mathbf{x}_i(t) = \int_0^t \mathbf{u}_i(\tau)\,d\tau \qquad i = 1, 2,$$

wobei $\mathbf{u}_i = \dot{\mathbf{x}}_i$ die Geschwindigkeit von Punkt x_i bezeichnet.
Die *relative Bewegungsbahn* $\mathbf{x}(t) := \mathbf{x}_1(t) - \mathbf{x}_2(t)$ der beiden Punkte genügt der Gleichung

$$\mathbf{x}(t) = \int_0^t \big(\mathbf{u}_1(\tau) - \mathbf{u}_2(\tau)\big)\,d\tau = \int_0^t \mathbf{u}(\tau)\,d\tau\,.$$

Der Vektor $\mathbf{u} := \mathbf{u}_1 - \mathbf{u}_2$ heißt *Relativgeschwindigkeit* von x_1 zu x_2. Die Länge der Bewegungskurve ist die sogenannte Bogenlänge $L_x(t)$, die sich folgendermaßen bestimmt:

$$L_x(t) := \int_0^t \|\dot{\mathbf{x}}(\tau)\|\,d\tau = \int_0^t \|\mathbf{u}(\tau)\|\,d\tau\,.$$

Wir wollen zunächst ein hinreichendes Kriterium für die Kollisionsfreiheit einer Bewegung innerhalb des Zeitintervalls $[0, T]$ herleiten.

Lemma 4.1. *Falls für zwei starre Körper $\mathcal{K}_1$, $\mathcal{K}_2$, die sich zum Zeitpunkt 0 nicht schneiden, die folgende Bedingung erfüllt ist:*

$$\max_{x_1 \in \mathcal{K}_1, x_2 \in \mathcal{K}_2} L_x(T) < \delta\big(\mathcal{K}_1(0), \mathcal{K}_2(0)\big)\,,$$

dann gilt für jeden Zeitpunkt $t \in [0, T]$:

$$\mathcal{K}_1(t) \cap \mathcal{K}_2(t) = \emptyset\,.$$

Beweis. Wir betrachten den Abstand zweier Punkte x_1, x_2 der beiden Körper zum Zeitpunkt $t \in [0, T]$

$$\begin{aligned}
\|\mathbf{x}_1(t) - \mathbf{x}_2(t)\| &= \|\mathbf{x}_1(t) - \mathbf{x}_1(0) + \mathbf{x}_1(0) - \mathbf{x}_2(0) + \mathbf{x}_2(0) - \mathbf{x}_2(t)\| \\
&\geq \|\mathbf{x}_1(0) - \mathbf{x}_2(0)\| - \|\mathbf{x}_1(t) - \mathbf{x}_1(0) + \mathbf{x}_2(0) - \mathbf{x}_2(t)\| \\
&= \delta\big(x_1(0), x_2(0)\big) - \|\mathbf{x}(t) - \mathbf{x}(0)\|\,.
\end{aligned}$$

Da die beiden Körper zum Zeitpunkt 0 kollisionsfrei waren, gilt $\delta\big(x_1(0), x_2(0)\big) > 0$. Aus der Voraussetzung des Lemmas folgt nun:

$$\begin{aligned}
\|\mathbf{x}(t) - \mathbf{x}(0)\| &= \left\| \int_0^t \dot{\mathbf{x}}(\tau)\,d\tau \right\| \\
&\leq \int_0^t \|\dot{\mathbf{x}}(\tau)\|\,d\tau \\
&= L_x(t) < \delta\big(\mathcal{K}_1(0), \mathcal{K}_2(0)\big) \leq \delta\big(x_1(0), x_2(0)\big)\,.
\end{aligned}$$

Es gilt daher $\|\mathbf{x}_1(t) - \mathbf{x}_2(t)\| > 0$ und somit $\mathcal{K}_1(t) \cap \mathcal{K}_2(t) = \emptyset$. □

Das Lemma zeigt eine Möglichkeit zur Berechnung unterer Kollisionszeitschranken auf. Sei (x_1, x_2) ein Punktepaar mit $x(t_c) = 0$, dann haben die beiden Punkte im Intervall $[0, t_c - t_0]$ einen Weg der Mindestlänge $\delta\big(\mathcal{K}_1(0), \mathcal{K}_2(0)\big)$ zurückgelegt. Ermittelt man nun den kleinsten Zeitschritt $t \in [0, T]$, in dem die relative Bahnkurve eines Punktepaares der beiden Körper diese Bogenlänge erreicht, so hat man eine untere Kollisionszeitschranke gefunden. Dabei ist selbstverständlich zu beachten, daß ein solcher Zeitschritt $t \in [0, T]$ nicht zwangsläufig existieren muß, da sich die Körper voneinander entfernen oder gar mit gleicher Geschwindigkeit in dieselbe Richtung bewegen können. Eine untere Kollisionszeitschranke t_L zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ für das Zeitintervall $[0, T]$ erfüllt daher die folgende Bedingung:

$$t_L \leq t_0 + \inf\left\{t \in [0, T] \,\middle|\, L_x(t) = \delta\big(\mathcal{K}_1(0), \mathcal{K}_2(0)\big),\, x_1 \in \mathcal{K}_1,\, x_2 \in \mathcal{K}_2\right\}.$$

Unter der Annahme ballistischer Bewegungsbahnen gilt für die Geschwindigkeit eines beliebigen Punktes x_i von Körper $\mathcal{K}_i$, $i = 1, 2$, aufgrund von 2.5 und 2.19:

$$\begin{aligned} \mathbf{u}_i(t) &= \boldsymbol{\nu}_i(t) + \boldsymbol{\omega}(t) \times \mathbf{r}_i(t) \\ &= \boldsymbol{\nu}_i(0) + \mathbf{g}t + \boldsymbol{\omega}_i(t) \times \mathbf{r}_i(t)\,. \end{aligned}$$

Dabei ist $\boldsymbol{\nu}_i$ die lineare Geschwindigkeit des Massenzentrums $\mathbf{c}_i$ und $\boldsymbol{\omega}_i$ die Winkelgeschwindigkeit von Körper $\mathcal{K}_i$, $i = 1, 2$. Des weiteren bezeichnet $\mathbf{r}_i := \mathbf{x}_i - \mathbf{c}_i$ den Vektor vom Massenzentrum zu dem Punkt x_i und $\mathbf{g} := (0, -g, 0)^T$, mit $g = 9.81$, den Beschleunigungsvektor. Für die Relativgeschwindigkeit der beiden Körper $\mathbf{u}$ ergibt sich somit:

$$\begin{aligned} \mathbf{u}(t) &= \boldsymbol{\nu}_1(t) - \boldsymbol{\nu}_2(t) + \boldsymbol{\omega}_1(t) \times \mathbf{r}_1(t) - \boldsymbol{\omega}_2(t) \times \mathbf{r}_2(t) \\ &= \boldsymbol{\nu}_1(0) - \boldsymbol{\nu}_2(0) + \boldsymbol{\omega}_1(t) \times \mathbf{r}_1(t) - \boldsymbol{\omega}_2(t) \times \mathbf{r}_2(t) \\ &= \boldsymbol{\nu}(0) + \boldsymbol{\omega}_1(t) \times \mathbf{r}_1(t) - \boldsymbol{\omega}_2(t) \times \mathbf{r}_2(t)\,. \end{aligned}$$

Unter der Annahme, daß $\|\boldsymbol{\omega}_i\|$ und $\|\mathbf{r}_i\|$ durch den Wert ω_i^{max} bzw. r_i^{max}, $i = 1, 2$, beschränkt werden können, läßt sich die Bogenlänge der Bewegungsbahn von x_1 und x_2 folgendermaßen nach oben abschätzen:

$$\begin{aligned} L_x(t) &= \int_0^t \|\mathbf{u}(\tau)\|\, d\tau \\ &\leq \int_0^t \big(\|\boldsymbol{\nu}(0)\| + \|\boldsymbol{\omega}_1(\tau) \times \mathbf{r}_1(\tau)\| + \|\boldsymbol{\omega}_2(\tau) \times \mathbf{r}_2(\tau)\|\big)\, d\tau \\ &\leq \int_0^t \big(\|\boldsymbol{\nu}(0)\| + \omega_1^{max} r_1^{max} + \omega_2^{max} r_2^{max}\big)\, d\tau \\ &= \big(\|\boldsymbol{\nu}(0)\| + \omega_1^{max} r_1^{max} + \omega_2^{max} r_2^{max}\big)t\,. \end{aligned} \tag{4.1}$$

Dabei haben wir verwendet, daß $\|\boldsymbol{\omega}_i(\tau) \times \mathbf{r}_i(\tau)\| \leq \|\boldsymbol{\omega}_i(\tau)\|\|\mathbf{r}_i(\tau)\| \sin\varphi \leq \omega_i^{max} r_i^{max}$, $i = 1, 2$, gilt.
Unterstellen wir weiter, daß die Bedingung $\|\mathbf{r}_i\| = \|\mathbf{x}_i - \mathbf{c}_i\| \leq r_i^{max}$ für alle Punkte $x_i \in \mathcal{K}_i$, i=1,2, erfüllt ist, so besitzt Abschätzung 4.1 Gültigkeit für alle Punktepaare

(x_1, x_2) der beiden Körper. Wir erhalten somit als untere Kollisionszeitschranke t_L den folgenden Wert:

$$t_L := t_0 + \begin{cases} \frac{\delta\left(\mathcal{K}_1(0), \mathcal{K}_2(0)\right)}{\|v(0)\| + \omega_1^{max} r_1^{max} + \omega_2^{max} r_2^{max}} & \text{falls } \|v(0)\| + \omega_1^{max} r_1^{max} + \omega_2^{max}\,; r_2^{max} > 0 \\ \infty & \text{sonst .} \end{cases}$$

Die maximale Länge r^{max} des Vektors $\mathbf{r}$ wird durch den Randpunkt des Körpers bestimmt, der die größte Entfernung zum Massenschwerpunkt c besitzt. Dieser Punkt kann bereits in der Vorberechnungsphase ermittelt werden. Als Körperradius erhält man somit:

$$r^{max} := \max_{v \in \mathcal{V}} \|\boldsymbol{v} - \mathbf{c}\| .$$

Die Bestimmung von ω^{max} erfordert dagegen weitergehende physikalische Überlegungen. In der Arbeit von MIRTICH [Mir96b] wird eine solche Schranke für die Winkelgeschwindigkeit mit Hilfe eines Energieerhaltungsargumentes hergeleitet. Die Betrachtung ballistischer Bahnen stellt nämlich sicher, daß die Rotationsenergie in einer solchen Bewegungsphase konstant ist. Es gilt daher:

$$E = \frac{1}{2}\boldsymbol{\omega}_{\mathcal{B}}(t)\mathbf{I}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}(t) = \text{const} ,$$

wobei das Superskript $\mathcal{B}$ anzeigt, daß die Trägheitsmatrix $\mathbf{I}_{\mathcal{B}}$ und der Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}_{\mathcal{B}} = (\omega_x, \omega_y, \omega_z)^T$ in Körperkoordinaten ausgedrückt sind.
Da $\mathbf{I}_{\mathcal{B}}$ eine Diagonalmatrix mit Einträgen I_x, I_y und I_z ist (vgl. Abschnitt 2.5.3), erhalten wir:

$$\begin{aligned} & I_x\omega_x(t)^2 + I_y\omega_y(t)^2 + I_z\omega_z(t)^2 = 2E \\ \Longrightarrow \quad & \min\left\{I_x, I_y, I_z\right\}\left(\omega_x(t)^2 + \omega_y(t)^2 + \omega_z(t)^2\right) \leq 2E \\ \Longleftrightarrow \quad & \|\boldsymbol{\omega}(t)\| \leq \sqrt{\frac{2E}{\min\{I_x, I_y, I_z\}}} . \end{aligned}$$

Diese Beobachtung liefert die gesuchte obere Schranke für die Winkelgeschwindigkeit während der ballistischen Bewegung:

$$\omega^{max} := \sqrt{\frac{2E}{\min\{I_x, I_y, I_z\}}} .$$

Die Überlegungen dieses Abschnittes führen zu Algorithmus 26, der eine untere Kollisionszeitschranke für die Bewegung zweier Körper während eines vorgegebenen Zeitintervalls ermittelt.

Algorithmus 26 Ein Algorithmus zur Bestimmung einer unteren Kollisionszeitschranke.

Eingabe: Die beiden Körper $\mathcal{K}_1$, $\mathcal{K}_2$, $d = \delta\big(\mathcal{K}_1(t_0), \mathcal{K}_2(t_0)\big)$ und das zugrunde liegende Zeitintervall $[t_0, t_1]$.
Ausgabe: Die untere Kollisionszeitschranke t_L.

```
TOI(d, (K1, K2), t0, t1)
(1)   if ‖v(t0)‖ + ω1^max r1^max + ω2^max r2^max > 0
(2)     tL ← t0 + d / (‖v(t0)‖ + ω1^max r1^max + ω2^max r2^max)
(3)     if tL ≤ t1
(4)       return tL
(5)   return ∞
```

4.3 Die Abtastung der Distanzfunktion mit Hilfe des Kollisionsheaps

4.3.1 Der Zweikörperfall

Unsere Überlegungen in den vorangegangenen Abschnitten erlauben, einen sehr einfachen Algorithmus zur Kollisionserkennung zweier Körper $\mathcal{K}_1$ und $\mathcal{K}_2$ zu formulieren. Wie wir bereits in 4.1 erläutert haben, stellt das Verfahren sicher, daß keine Kollision übersehen wird, da die frühste Nullstelle t_c der Distanzfunktion ausgehend von einer kollisionsfreien Konfiguration der beiden Körper mit Hilfe unterer Kollisionszeitschranken approximiert wird. Dazu wertet die Abstandsberechnung die Distanzfunktion der beiden Körper zum aktuellen Zeitpunkt t_i aus. Falls der ermittelte Abstand $\delta\big(\mathcal{K}_1(t_i), \mathcal{K}_2(t_i)\big)$ die Kollisionstoleranz ε_c noch nicht unterschritten hat, bestimmen wir ausgehend von den aktuellen Dynamikdaten die untere Kollisionszeitschranke t_{i+1}. Diese identifiziert den frühsten Zeitpunkt, an dem eine Kollision zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ möglich ist. Wir können nun den Dynamikzustand des Systems bis zum Zeitpunkt t_{i+1} fortschreiben und dabei sicher sein, daß die Bewegungsbahnen der beiden Körper im Intervall $[t_i, t_{i+1}]$ kollisionsfrei sind. Zum Zeitpunkt t_{i+1} kann diese Garantie nicht mehr gegeben werden, so daß eine erneute Auswertung der Distanzfunktion erforderlich ist.

Auf diese Weise tastet sich das Kollisionserkennungssystem immer dichter an den tatsächlichen Kollisionszeitpunkt t_c heran. Wird schließlich der Schwellenwert ε_c von dem aktuellen Abstandswert unterschritten, so liefert das System den aktuellen Zeitpunkt als approximativen Kollisionszeitpunkt $\hat{t}_c$ zurück und löst die Kollisionsauflösung aus. Die von ihr berechneten Kollisionsimpulse verhindern eine Durchdringung der beiden Körper, indem sie die Geschwindigkeiten der beteiligten Objekte unmittelbar ändern und somit eine neue ballistische Bewegungsphase einleiten. Die Länge dieser Phase muß dabei durch eine erneute Abtastung der Distanzfunktion bestimmt werden. Algorithmus 27 konkretisiert diese Darstellung.

Algorithmus 27 Ein Algorithmus zur Approximation der frühsten Nullstelle der Distanzfunktion.

Eingabe: Die beiden Körper $\mathcal{K}_1$, $\mathcal{K}_2$ und das Zeitintervall $[t_0, t_1]$, das auf eine Kollision von $\mathcal{K}_1$ und $\mathcal{K}_2$ hin untersucht werden soll.
Ausgabe: $\hat{t}_c = \inf\left\{t \in [t_0, t_1] \,\middle|\, 0 \leq d(\mathcal{K}_1, \mathcal{K}_2, t) < \varepsilon_c\right\}$.

DISTFUNCROOT$((\mathcal{K}_1, \mathcal{K}_2), t_0, t_1)$
(1) **if** $t_1 < t_0$
(2) **return** ∞
(3) $d \leftarrow \delta\big(\mathcal{K}_1(t_0), \mathcal{K}_2(t_0)\big)$
(4) **while** $d > \varepsilon_c$
(5) $t_L \leftarrow \text{TOI}(d, (\mathcal{K}_1, \mathcal{K}_2), t_0, t_1)$
(6) $d \leftarrow \delta\big(\mathcal{K}_1(t_L), \mathcal{K}_2(t_L)\big)$
(7) **return** t_L

Das vorgestellte Schema steht im Einklang mit der Intuition, daß bei geringer werdendem Abstand auch „vorsichtiger" agiert werden muß. Solange die Objekte weit voneinander entfernt sind, können große Zeitintervalle ohne Abstandstests überbrückt werden. Erst wenn die Objekte sich einander nähern, häufen sich die durchzuführenden Abstandsberechnungen.

4.3.2 Der n-Körperfall

Geht man nun von dem Problem der Kollisionserkennung zweier Körper zum allgemeinen Fall, dem sogenannten *n-body-collision-detection problem* über, so ist ein Scheduling der $O(n^2)$ Abstandstests entsprechend ihrer Kollisionszeitschranken erforderlich. Mit Hilfe einer Prioritätswarteschlange, die alle Objektpaare entsprechend ihres frühsten Kollisionszeitpunktes ordnet, kann auf einfache Weise der nächste durchzuführende Abstandstest ermittelt werden. Die Sortierung stellt dabei sicher, daß der Eintrag der Prioritätswarteschlange mit kleinstem Schlüssel das Körperpaar mit frühstem potentiellen Kollisionszeitpunkt identifiziert. Aufgrund ihrer Implementierung bezeichnen wir die Datenstruktur im folgenden als *Kollisionsheap*.

In einem Kollisionserkennungsschritt wird nun der Eintrag mit kleinstem Schlüssel abgespalten und das entsprechende Körperpaar zur Abstandsberechnung weitergereicht. Ist der ermittelte Distanzwert größer ε_c, so war die Kollisionszeitschranke nicht scharf genug. Anhand der aktuellen Abstands- und Dynamikdaten kann der frühste Kollisionszeitpunkt korrigiert und das Körperpaar mit neuem Schlüssel in den Kollisionsheap eingefügt werden. Falls der Abstand dagegen die ε_c-Toleranz unterschreitet, so führen die von der Kollisionsauflösung ermittelten Kollisionsimpulse zu einer unmittelbaren Änderung der kinematischen Eigenschaften der beiden Kollisionspartner. Dies bedeutet jedoch, daß die Kollisionszeitschranken aller Körperpaare, an denen die kollidierenden Objekte beteiligt sind, ihre Gültigkeit verloren haben. Wir erinnern uns, daß die Berechnung des frühsten Kollisionszeitpunktes auf der Annahme ballistischer

und somit kollisionsfreier Bewegungsbahnen beruhen.

Eine Kollision löst daher $O(n)$ Abstandsberechnungen zur Aktualisierung des Kollisionsheaps aus, wie aus Algorithmus 28 ersichtlich wird. H bezeichnet den Kollisionsheap, auf dem die in Abschnitt 3.8.4 vorgestellten Operationen einer Prioritätswarteschlange durchgeführt werden können.

Algorithmus 28 Die Aktualisierung der Heapeinträge nach einer Kollision.

Eingabe: Die kollidierten Körper $\mathcal{K}_1$, $\mathcal{K}_2$ sowie das zugrunde liegende Zeitintervall $[t_0, t_1]$.

UPDATEHEAP$((\mathcal{K}_1, \mathcal{K}_2), t_0, t_1)$

(1) **foreach** $\left[t, (\mathcal{K}_1, \mathcal{K})\right] \in H$
(2) $\quad d \leftarrow \delta(\mathcal{K}_1, \mathcal{K})$
(3) $\quad t_L \leftarrow \text{TOI}(d, (\mathcal{K}_1, \mathcal{K}), t_0, t_1)$
(4) $\quad$ H.CHANGEKEY$(\left[t, (\mathcal{K}_1, \mathcal{K})\right], t_L)$
(5) **foreach** $\left[t, (\mathcal{K}_2, \mathcal{K})\right] \in H$
(6) $\quad d \leftarrow \delta(\mathcal{K}_2, \mathcal{K})$
(7) $\quad t_L \leftarrow \text{TOI}(d, (\mathcal{K}_2, \mathcal{K}), t_0, t_1)$
(8) $\quad$ H.CHANGEKEY$([t, (\mathcal{K}_2, \mathcal{K})], t_L)$
(9) $d \leftarrow \delta(\mathcal{K}_1, \mathcal{K}_2)$
(10) $t_L \leftarrow \text{TOI}(d, (\mathcal{K}_1, \mathcal{K}_2), t_0, t_1)$
(11) H.INSERT$(t_L, (\mathcal{K}_1, \mathcal{K}_2))$

Dieser Aufwand kann jedoch unter Ausnutzung *räumlicher Kohärenz* reduziert werden, wie wir im nächsten Abschnitt sehen werden.

4.3.3 Die Aktualisierung des Kollisionsheaps unter Ausnutzung räumlicher Kohärenz

Es gestaltet sich einfach, Beispiele zu konstruieren, in denen das naive $O(n)$-Aktualisierungsverfahren die Kollisionserkennung in die Knie zwingt. Läßt man beispielsweise eine Menge von Münzen auf eine Ebene fallen, so steigt die Zahl der aufzulösenden Kollisionen drastisch an, wenn die Münzen beginnen, ihre endgültige Lage auf der Ebene einzunehmen. Obwohl die Münzen ausreichend weit voneinander entfernt sind, muß nach jeder Kollision der Abstand der beiden Kollisionspartner zu allen anderen Münzen neu berechnet werden.

An dieser Stelle wird die Notwendigkeit der Ausnutzung räumlicher Beziehungen offensichtlich. Kollisionszeitschranken sollten lediglich zwischen solchen Körperpaaren aktualisiert werden, deren räumliche Nähe eine Kollision innerhalb des nächsten zu betrachtenden Zeitintervalls ermöglicht. Diese Idee wollen wir im folgenden präzisieren.

4.3.4 Lokalisation der Bewegungshüllkörper zur Identifkation naher Objektpaare

Um die Zahl der Aktualisierungen und Abstandstests zwischen weit entfernten Körperpaaren zu reduzieren, sollten nur solche Paare im Kollisionsheap gespeichert sein, deren Bewegungsbahnen sich im zu betrachtenden Zeitintervall nahe kommen. Um diese räumlichen Beziehungen berücksichtigen zu können, bietet es sich an, die aufgrund der Bewegung der Objekte überstrichenen Volumina (*Swept-Volumes*) miteinander zu vergleichen. Eine einfache, jedoch sehr effiziente Methode besteht nun darin, diese Swept-Volumes durch Hüllkörper zu überdecken und letztere im Raum zu lokalisieren. Objekte sind in diesem Sinne innerhalb eines Zeitschritts „nahe“, falls der Abstand der beiden *Bewegungshüllkörper* einen Schwellenwert unterschreitet. Die notwendigen algorithmischen Voraussetzungen hierzu wurden bereits in Kapitel 3.7.6 gelegt.

In der Arbeit von LIN [Lin93] und MIRTICH [Mir97a] werden ähnliche Verfahren zur räumlichen Lokalisation der Bewegungsbahnen diskutiert. Auch sie beruhen auf der Idee, Bewegungshüllkörper auf räumliche Nähe zu testen. Als Hüllkörpertyp werden achsenorientierte Boxen (Iso-Boxen) eingesetzt, die die Objekte im betrachteten Zeitintervall einschließen. Paare von Iso-Boxen, die überlappen, bestimmen die räumlich benachbarten Objektpaare. Die Verfahren mit denen diese Überlappungen identifiziert werden, wollen wir in Abschnitten 4.4.2 bis 4.4.6 beschreiben.

4.3.5 Einfügen und Löschen von nahen Objektpaaren

Da wir nicht mehr alle Objektpaare im Kollisionsheap verwalten wollen, sondern lediglich solche, die sich im nächsten Simulationsschritt so nahe kommen, daß eine Kollision nicht ausgeschlossen werden kann, müssen wir entscheiden, welche Paare in den Heap aufzunehmen sind und welche aus diesem entfernt werden können. Hat die Simulation das dynamische System bis zum Zeitpunkt t_0 entwickelt, so liefert die Kollisionszeitschranke t_L des obersten Heapelementes die *maximale* Länge $t_L - t_0$ des nächsten Simulationsschrittes. Im Gegensatz zur bisherigen Vorgehensweise, in der dieser Wert die exakte Länge des nächsten Zeitschrittes definiert hat, können Objektpaare die erst für das Intervall $[t_0, t_L]$ in den Heap aufgenommen werden, den frühsten Kollisionszeitpunkt zeitlich nach vorne verlegen. Dies wollen wir im folgenden erläutern. Mit Hilfe der Swept-Volume-Technik ist es zunächst möglich, potentielle Kollisionskandidaten für das Intervall $[t_0, t_L]$ zu identifizieren. Da es sich dabei um Paare handelt, die im nächsten Zeitschritt von dem Kollisionserkennungssystem beachtet werden müssen, berechnen wir die jeweilige untere Kollisionszeitschranke und fügen das Objektpaar unter diesem Schlüssel in den Heap ein. Es ist jedoch zu beachten, daß die Einfügeoperation dem eigentlichen Simulationschritt, d.h. der Integration der ballistischen Bewegungsbahnen vorausgehen muß. Der Grund hierfür ist, daß möglicherweise ein neu hinzukommender Kollisionskandidat das oberste Heapelement verdrängt und das dynamische System daher nur bis zu dessen Kollisionszeitschranke $t'_L < t_L$ entwickelt werden darf. Die Länge des eigentlichen Simulationsschrittes kann also erst nach dem

Einfügen aller potentiellen Kollisionspaare festgelegt werden.

Da Objektpaare in den Kollisionsheap aufgenommen werden, wenn sie einander nahe kommen, sollten sie selbstverständlich auch wieder aus diesem gelöscht werden, sobald sie sich ausreichend weit voneinander entfernt haben. Um zu verhindern, daß Objektpaare nach ihrer Entfernung aus dem Heap sofort wieder aufgenommen werden müssen, schlägt MIRTICH ein *Hysteresis*-Konzept vor. Ein Kollisionskandidat verbleibt solange im Heap, bis er als oberstes Heapelement zur Abstandsberechnung herangezogen wird. Mit Hilfe des Tests auf räumliche Nähe überprüft man nun, ob das Objekt mit einer korrigierten Kollisionszeitschranke im Heap behalten werden soll. Sind die beiden Körper im nächsten Zeitintervall nicht mehr nahe, so wird das Paar endgültig aus dem Kollisionsheap entfernt.

Der Einsatz dieser Technik reduziert die Zahl der Objektpaare, die im Kollisionsheap verwaltet und somit der Abstandsberechnung in unregelmäßigen Zeitschritten übergeben werden auf diejenigen, die aufgrund ihrer Lage im Raum tatsächlich als Kollisionskandidaten in Frage kommen. Somit werden nach einer Kollision lediglich die Kollisionszeitschranken zwischen den Körperpaaren aktualisiert, für die im nächsten Zeitschritt eine Kollision nicht ausgeschlossen werden können.

4.4 Bestimmung überlappender Bewegungshüllkörper

Um auf effiziente Weise Aussagen über die räumliche Nähe der Objektpaare innerhalb eines Simulationsschrittes machen zu können, wollen wir die durch die Bewegung der Objekte überstrichenen Volumina in Hüllkörper einschließen. Als Hüllkörpertyp können prinzipiell alle in Abschnitt 3.5.2 vorgestellten Primitive eingesetzt werden. Im folgenden werden wir jedoch sehen, daß sich Iso-Boxen aufgrund der Einfachheit ihrer Konstruktion sowie effizienter Überlappungstests für diese Aufgabe empfehlen.

4.4.1 Die Konstruktion achsenorientierter Bewegungshüllkörper

Das Zeitintervall, das der Betrachtung zugrunde liegt, wird durch den aktuellen Zeitpunkt t_0 und die Kollisionszeitschranke des obersten Kollisionsheapeintrags bestimmt. Ausgehend von der geometrischen Lage und den Dynamikdaten eines Objektes zum Zeitpunkt t_0 ist die Iso-Box zu bestimmen, die das überstrichene Volumen des Objektes im Intervall $[t_0, t_L]$ umhüllt. Da die vollständige Neuberechnung einer solchen achsenorientierten Box für jedes Objekt nach jedem durchgeführten Simulationsschritt unumgänglich ist, muß der Aufwand zur Konstruktion des Bewegungshüllkörpers minimal sein.

Betrachtet man lediglich den Massenschwerpunkt c des Körpers, so läuft dieser unter Annahme einer ballistischen Bewegung im Intervall $[t_0, t_L]$ auf einer Parabelbahn. Diese genügt der Gleichung

$$\mathbf{c}(t) = \mathbf{c}(t_0) + \boldsymbol{\nu}(t_0)t + \frac{1}{2}\mathbf{a}t^2\,, \quad t \in [t_0, t_L]\,,$$

wobei $\mathbf{a} = (0, -g, 0)^T$ den Beschleunigungsvektor bezeichnet. Um die Rotationsbewegung des Körpers berücksichtigen zu können, legen wir eine Kugel mit Radius r^{max} um den Massenschwerpunkt. Dabei ist r^{max}, wie in 4.2 definiert, die Länge des Vektors von $\mathbf{c}$ zu dem am weitesten entfernten Randpunkt des Körpers. Um die Bewegungsbahn dieser Kugel und damit die des gesamten Objektes im zugrundeliegenden Zeitintervall einschließen zu können, sind maximal drei Punkte zu berücksichtigen.

1. Die Lage des Massenschwerpunktes zu Beginn der Bewegung: $\mathbf{c}(t_0)$.
2. Die Endlage des Massenschwerpunktes: $\mathbf{c}(t_L)$.
3. Der Hochpunkt der Parabelbahn in der der Gravitationswirkung entgegengesetzten Richtung: $\mathbf{c}(v_2(t_0)/g)$, falls $v_2(t_0)/g \in [t_0, t_L]$.

Die Koordinatenintervalle $[d_i^{min}, d_i^{max}]$, $1 \leq i \leq 3$, der gesuchten Iso-Box können also folgendermaßen bestimmt werden:

$$
\begin{aligned}
d_i^{min} &= -r^{max} + \min\left\{c_i(t_0),\, c_i(t_L)\right\} \\
d_i^{max} &= r^{max} + \begin{cases} \max\left\{c_i(t_0),\, c_i(t_L)\right\} & \text{falls } i \neq 2 \vee v_2(t_0)/g \notin [t_0, t_L]\,; \\ \max\left\{c_i(t_0),\, c_i(t_L),\, c_i(v_2(t_0)/g)\right\} & \text{sonst}\,. \end{cases}
\end{aligned}
$$

Im Falle statischer bzw. ausschließlich translatorisch bewegter Objekte kann der Bewegungshüllkörper präziser berechnet werden, da hier keine Rotationsbewegung beachtet werden muß.

4.4.2 Das naive Verfahren

Die Objektpaare, die innerhalb des betrachteten Zeitfensters als mögliche Kollisionspaare zu berücksichtigen sind, müssen durch einen Überlappungstest der Bewegungshüllkörper identifiziert werden. Da wir uns für achsenorientierte Boxen als Hüllkörpertyp entschieden haben, besteht die Aufgabe nun darin, alle Paare von sich schneidender Iso-Boxen zu bestimmen. Es erscheint daher sinnvoll das Problem zunächst für ein einzelnes Paar von Boxen zu betrachten. Eine Iso-Box $A = AABB(\mathbf{d}^{min}, \mathbf{d}^{max})$, die auf die drei Koordinatenachsen projiziert wird, liefert auf jeder Achse ein Intervall, das die Ausdehnung der Box entlang dieser Achse angibt:

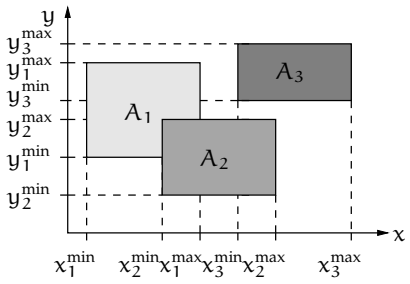
$$A = X \times Y \times Z\,, \quad \text{mit } X = [d_1^{min}, d_1^{max}], \quad Y = [d_2^{min}, d_2^{max}], \quad Z = [d_3^{min}, d_3^{max}]\,.$$

Für den Überlappungstest erweist sich die folgende Beobachtung als hilfreich.

Beobachtung 6. Zwei achsenorientierte Boxen $A_i = AABB(\mathbf{d}_i^{min}, \mathbf{d}_i^{max})$, $i = 1, 2$ schneiden sich genau dann, wenn ihre Projektion auf allen drei Koordinatenachsen überlappen:

$$
\begin{aligned}
& A_1 \cap A_2 \neq \emptyset \\
\iff\ & X_1 \cap X_2 \neq \emptyset \ \wedge\ Y_1 \cap Y_2 \neq \emptyset \ \wedge\ Z_1 \cap Z_2 \neq \emptyset \\
\iff\ & \forall j, 1 \leq j \leq 3:\ d_{1j}^{min} \in [d_{2j}^{min}, d_{2j}^{max}] \ \vee\ d_{2j}^{min} \in [d_{1j}^{min}, d_{1j}^{max}] \\
\iff\ & \forall j, 1 \leq j \leq 3:\ d_{2j}^{min} \leq d_{1j}^{min} \leq d_{2j}^{max} \ \vee\ d_{1j}^{min} \leq d_{2j}^{min} \leq d_{1j}^{max}\,.
\end{aligned}
$$

Abbildung 4.2 Das 1D-Sweep-and-Prune-Verfahren.



$$L_x = (d_{11}^{min}, d_{21}^{min}, d_{11}^{max}, d_{31}^{min}, d_{21}^{max}, d_{31}^{max})$$

$$L_y = (d_{22}^{min}, d_{12}^{min}, d_{22}^{max}, d_{32}^{min}, d_{12}^{max}, d_{32}^{max})$$

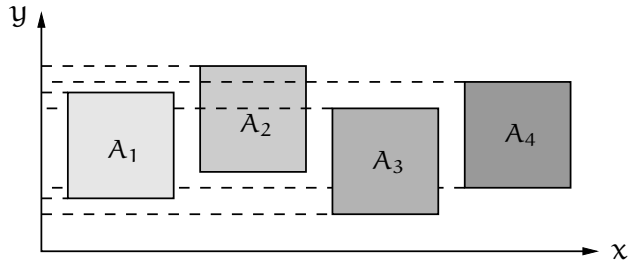
Setzt sich die zugrundeliegende Szene aus n Objekten zusammen, so überprüft der naive Algorithmus alle Paare von achsenorientierten Boxen (A_i, A_j), $1 \leq i < j \leq n$ auf Überlappung. Unsere Beobachtung liefert dazu einen effizienten elementaren Schnittest, mit dessen Hilfe alle Paare sich schneidender Bewegungshüllkörper in Zeit $O(n^2)$ identifiziert werden können.

4.4.3 Das 1D-Sweep-and-Prune-Verfahren

Im folgenden werden wir einen weiteren Algorithmus vorstellen, der das Überlappungsproblem auf eine Fragestellung im eindimensionalen Raum reduziert. Das sogenannte 1D-*Sweep-and-Prune*-Verfahren von LIN [Lin93] findet die gesuchten Paare sich schneidender Iso-Boxen anhand überlappender Intervalle auf allen drei Koordinatenachsen in „erwarteter" Linearzeit. Während das naive Verfahren den Intervalltest für jedes AABB-Paar durchführt, reduziert der Algorithmus von LIN die Zahl der Tests durch eine gleichzeitige Betrachtung aller AABB-Projektionen auf die einzelnen Koordinatenachsen. Das Verfahren verwendet dazu drei Listen L_x, L_y und L_z für jede Achse und speichert in diesen die entsprechenden Intervallgrenzen der Iso-Boxen $A_1, \ldots, A_n$ ab. Anschließend werden die *Koordinatenlisten* aufsteigend sortiert, so daß die überlappenden Paare von Iso-Boxen durch einmaliges Traversieren der Listen ermittelt werden können. Abbildung 4.2 veranschaulicht das Konzept für den zweidimensionalen Fall. In dem Beispiel sind die Listen L_x und L_y bereits sortiert, so daß die Intervallüberlappungen auf der x- und y-Achse abgelesen werden können. Offensichtlich liegt nur zwischen A_1 und A_2 ein nichtleerer Schnitt vor. Die Laufzeit des Verfahrens beträgt $O(n \log n + c)$, wobei c die Zahl der überlappenden Boxenpaare bezeichnet.

Unterstellt man temporäre Kohärenz aufgrund kleiner Objektbewegungen in zwei aufeinanderfolgenden Zeitschritten, so werden sich die entsprechenden Koordinatenlisten nur geringfügig unterscheiden. Eine solche annähernd sortierte Liste kann mit Hilfe von INSERTION SORT in Zeit $O(e)$ aktualisiert werden. Dabei gibt e die Zahl der

Abbildung 4.3 Eine ungünstige Situation für das 1D-Sweep-and-Prune-Verfahen: Da die Iso-Boxen auf der y-Achse sehr dicht zusammenliegen, können die Sortierungskosten der Koordinatenlisten die worst-case-Komplexität erreichen.



Vertauschungen an, die im Rahmen des Sortierprozesses in der Liste L durchgeführt werden. Aufgrund unserer Kohärenzannahme können wir e_x, e_y und e_z vernachlässigen und eine lineare Laufzeit des Verfahrens erwarten. Ein weiterer positiver Kohärenzeffekt bei der Verwendung von INSERTION SORT ist die Möglichkeit, die Menge der Überlappungspaare des letzten Zeitschrittes im Rahmen des Sortierprozesses anzupassen statt neu zu berechnen. Die gleichzeitige Aktualisierung und Sortierung einer Liste geschieht dabei nach dem folgenden Schema. Ist die untere Intervallgrenze kleiner als die im letzten Zeitschritt, so führt lediglich eine Vertauschung mit einer oberen Intervallgrenze zu einem neuen Überlappungspaar. Wird die Intervallgrenze dagegen zum Ende der Liste hin verschoben, so ändert sich der Überlappungsstatus in „überlappungsfrei“, wenn eine Vertauschung mit einer oberen Intervallgrenze stattfindet. Die Menge der Überlappungspaare wird also um das an der Intervallvertauschung beteiligte Objektpaar reduziert. In Analogie zur Modifikation unterer Intervallgrenzen ändert sich im Rahmen der Aktualisierung einer oberen Intervallgrenze der Überlappungsstatus nur bei solchen Vertauschungsoperationen, an denen untere Intervallgrenzen beteiligt sind. Bei den Aktualisierungsoperationen ist die Reihenfolge der Grenzenmodifikationen zu beachten. Die Korrektheit der gerade beschriebenen Operationen zur Anpassung der Überlappungslisten ist nur dann gegeben, wenn stets die Bedingung erfüllt ist, daß untere und obere Grenze ein nichtleeres Intervall bilden.

Offensichtlich eignet sich der Algorithmus besonders gut für Szenarien, in denen die Iso-Boxen bezüglich der drei Koordinatenachsen gestreut liegen. Unterscheiden sich dagegen die minimalen und maximalen Koordinaten aller AABBs entlang einer Achse nur unwesentlich, so können die Sortierungskosten den worst-case von $O(n^2)$ tatsächlich erreichen, wie das Beispiel in Abbildung 4.3 andeutet. Aufgrund der extremen Dimensionsreduktion tritt dieses Problem selbst dann auf, wenn die Boxen entlang der übrigen Achsen gestreut liegen. Projiziert man die Iso-Boxen stattdessen auf die Koordinatenebenen, so kann zumindest in letzterem Fall das worst-case-Verhalten verhindert werden. Eine solche Vorgehensweise wollen wir im nächsten Abschnitt beschreiben.

4.4.4 Das 2D-Sweep-and-Prune-Verfahren

Führt man das Überlappungsproblem auf den zweidimensionalen Fall zurück, so ist eine Identifikation aller Boxenprojektionen erforderlich, die sich in zwei der drei Koordinatenebenen schneiden (*2D-Sweep-and-Prune*). Mit Hilfe eines *Sweep*-Verfahrens können alle überlappenden Iso-Boxen in Zeit $O(n \log n + K)$ bestimmt werden, wobei K die Zahl der überlappenden Rechtecke bezeichnet.

Projiziert man eine achsenorientierte Box auf eine der drei Koordinatenebenen, so erhält man ein achsenorientiertes Rechteck. Zwei Iso-Boxen schneiden sich genau dann, wenn sich ihre Projektionen in zwei Koordinatenebenen (o.B.d.A. in der xy- und xz-Ebene) überlappen.

Beobachtung 7. Seien $A_i = AABB(\mathbf{d}_i^{min}, \mathbf{d}_i^{max})$, $i = 1, 2$, zwei Iso-Boxen, so gilt:

$$A_1 \cap A_2 \neq \emptyset \quad \Longleftrightarrow \quad \pi_{xy}(A_1) \cap \pi_{xy}(A_2) \neq \emptyset \ \wedge \ \pi_{xz}(A_1) \cap \pi_{xz}(A_2) \neq \emptyset \, .$$

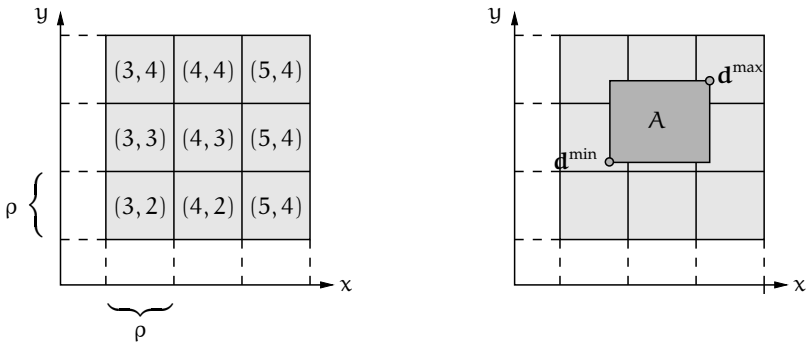
Um die sich schneidenden Rechtecke effizient identifizieren zu können, verwendet LIN [Lin93] einen *Intervallbaum*, der alle Überlappungen eines Anfrageintervalls in Zeit $O(n \log n + k)$ bestimmt, wobei k die Zahl der gefundenen Überlappungen beschreibt. Das Verfahren geht nun folgendermaßen vor. Im Falle einer Projektion auf die xy-Ebene wählt man zunächst eine Sweep-Linie parallel zur y-Achse, die sich in Richtung steigender x-Werte verschiebt. Die nichtfallend sortierten Projektionen der Iso-Boxen auf die x-Achse liefern die Werte, an denen die Sweep-Linie anhält. Trifft man auf das erste (bezüglich des Sweeps) y-Intervall $I_1^y(R_1)$ eines Rechtecks R_1, so entsprechen die y-Intervalle $I^y(R_2)$, die vom Intervallbaum als mit $I_1^y(R_1)$ überlappend gemeldet werden, Paaren von Rechtecken (R_1, R_2), die sich in der xy-Ebene schneiden. Das Intervall $I_1^y(R_1)$ wird anschließend in den Intervallbaum eingefügt. Erreicht die Sweep-Linie schließlich das zweite y-Intervall $I_2^y(R_1)$ von R_1, so muß das korrespondierende Intervall $I_1^y(R_1)$ aus dem Baum entfernt werden. Für den Überlappungstest zwischen n Iso-Boxen ergibt sich somit eine Laufzeit von $O(n \log n + K)$, wobei K die Gesamtzahl der überlappenden Rechtecke bezeichnet.

4.4.5 Das statische Raumpartitionierungsverfahren

Eine andere Vorgehensweise zur Bestimmung der überlappenden Paare von Bewegungshüllkörpern wird von MIRTICH in [Mir97a] vorgestellt. Das Verfahren basiert auf einer Arbeit von OVERMARS zur Lösung von Punktlokalisationsproblemen [Ove92]. Mit Hilfe einer *Raumpartitionierungstechnik* können die gesuchten Iso-Boxen-Paare in erwarteter Zeit $O(n + c)$ identifiziert werden, wobei c die Outputsensitivität des Algorithmus' wiederspiegelt (s.u.).

Das Prinzip der Raumpartitionierung findet in vielen Kollisionserkennungsansätzen Anwendung, um die „all-pairs weakness", die sich aus der notwendigen Betrachtung aller Objektpaare ergibt, in der Praxis zu mildern. Im Rahmen dieser Technik wird der Raum, der die zugrunde liegende Szene enthält, in sogenannte *Voxels* (für *volume elements*) aufgeteilt. Ein Voxel oder Raumwürfel ist ein quaderförmiges Raumelement, das nicht weiter unterteilt wird. Wie der Name Raumpartitionierung bereits andeutet,

Abbildung 4.4 Lokalisation von Iso-Boxen mit Hilfe einer uniformen Raumpartitionierung.



Ausschnitt einer Raumpartitionierung des $\mathbb{R}^2$ mit Gitterweite ρ.

Bestimmung der von der Iso-Box A geschnittenen Raumwürfel.

überlappen sich die Voxel nicht und ihre Vereinigung liefert den gesamten Raumbereich. Man unterscheidet im wesentlichen zwischen zwei Konzepten der Raumpartitionierung: Techniken, die ausschließlich Raumwürfel gleicher Größe erlauben (*uniforme Raumpartitionierung*) sowie Raumunterteilungen, deren Voxel in ihrer Größe variieren können. Das von MIRTICH modifizierte OVERMARS-Verfahren beruht auf uniformen Raumaufteilungen, die in einer Hierarchie zunehmend verfeinerter Auflösungen abgelegt sind. Auf diese Weise kann die Inflexibilität der uniformen Raumpartitionierung beseitigt werden.

Definition 4.1. Unter einer Raumpartitionierung mit Gitterweite ρ verstehen wir die folgende Abbildung, ξ_ρ, die jeden Punkt $\mathbf{x}$ des $\mathbb{R}^3$ einen durch das Tripel $\boldsymbol{w} = (w_1, w_2, w_3) \in \mathbb{Z}^3$ beschriebenen Raumwürfel zuordnet:

$$\begin{array}{rccc} \xi: & \mathbb{R}^3 & \rightarrow & \mathbb{Z}^3 \\ & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} & \mapsto & \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \lfloor x_1/\rho \rfloor \\ \lfloor x_2/\rho \rfloor \\ \lfloor x_3/\rho \rfloor \end{bmatrix} \end{array}$$

□

Ein Ausschnitt einer Raumpartitionierung ist im linken Teil von Abbildung 4.4 dargestellt. Die Raumwürfel, die von einer achsenorientierten Box $A = \mathrm{AABB}(\mathbf{d}^{min}, \mathbf{d}^{max})$ geschnitten werden, erhält man, indem man den Raumwürfel des Eckpunkts mit minimalen Koordinaten mit dem Voxel des Eckpunkts mit maximalem x-, y- und z-Wert vergleicht.

Beobachtung 8. Sei ξ_ρ die Abbildung, die die Raumpartitionierung der Gitterweite ρ definiert, und seien $\boldsymbol{w}^{min} = \xi_\rho(\mathbf{d}^{min})$ sowie $\boldsymbol{w}^{max} = \xi_\rho(\mathbf{d}^{max})$ die Raumwürfel, in denen

die Eckpunkte der Iso-Box A mit minimalen bzw. maximalen Koordinaten liegen. Dann ergibt sich die von A geschnittene Voxelmenge als

$$W_\rho(A) := \left\{ \boldsymbol{w} \in \mathbb{Z}^3 \,\middle|\, \forall i, 1 \leq i \leq 3 : \; w_i^{\min} \leq w_i \leq w_i^{\max} \right\} .$$

Der rechte Teil von Abbildung 4.4 verdeutlicht die Beobachtung für den zweidimensionalen Fall.

Die Verwendung einer *Hashtabelle* gestattet nun auf effiziente Weise die Boxenpaare zu identifizieren, die als Überlappungskandidaten in Frage kommen. In dieser Tabelle werden alle Raumwürfel repräsentiert, die von mindestens einer Box geschnitten werden. Dies sind lediglich endlich viele. Als Schlüssel verwendet man dabei das Tripel $\boldsymbol{w}$, welches den geschnittenen Raumwürfel charakterisiert. Wir nehmen an, daß für jede achsenorientierte Box A ein Verweis auf A in den durch $W(A)$ indizierten Buckets der Hashtabelle abgelegt ist. Betrachtet man nun die von einer beliebigen Anfragebox $\overline{A}$ adressierten Hashbuckets, so entspricht die Menge der darin „gespeicherten" Boxen mit gleichem Schlüssel wie $\overline{A}$ den Iso-Boxen, die sich einen Raumwürfel mit $\overline{A}$ teilen. Eine endgültige Entscheidung, ob diese Boxen die Anfragebox schneiden, liefert der im Rahmen des naiven Verfahrens (vgl. Abschnitt 4.4.2) vorgestellte Überlappungstest für AABBs.

Als *Hashfunktion* haben wir in unserer Implementierung eine Darstellung des Tripels $\boldsymbol{W}$ zur Basis 1000 gewählt, die mittels einer mod-Operation auf den Adreßbereich der Tabelle reduziert wird. Die Hashtabelle habe dabei die Größe einer Primzahl p.

$$\begin{aligned} h : \mathbb{Z}^3 &\to \{0, \ldots, p-1\} \\ \boldsymbol{w} &\mapsto \Big(\sum_{i=1}^{3} 1000^{i-1} w_i \Big) \bmod p \, . \end{aligned}$$

Abbildung 4.5 verdeutlicht die Arbeitsweise der Hashtabelle. Die Paare (B, C) und (C, D) werden durch die Tabelle als potentielle Überlappungspaare identifiziert. Erst durch den Schnittest zweier AABBs wird erkannt, daß B und C sich tatsächlich schneiden, während C und D überlappungsfrei sind.

Der Erfolg des Verfahrens hängt sehr stark von der Wahl der Gitterweite ρ ab. Dabei ist der folgende *Trade-Off* zu optimieren:

- Ist die Gitterweite klein, so sind die Aussagen über mögliche Überlappungen präzise. Auf der anderen Seite werden größere AABBs zahlreiche Raumwürfel schneiden und somit in vielen Buckets gespeichert. Dies wirkt sich insbesondere in einem dynamischen Umfeld negativ aus, da nach jedem Zeitschritt die Lage der Box innerhalb der Raumaufteilung aktualisiert werden muß.

- Wählt man die Gitterweite dagegen sehr groß, so ist die Raumpartitionierung eventuell nicht fein genug granuliert, um einen Großteil der Boxenpaare als schnittfrei zu klassifizieren. Es müssen daher zu viele Paare von AABBs mittels elementarer Schnittests auf Überlappung geprüft werden.

Abbildung 4.5 Lokalisation von Iso-Boxen mit Hilfe einer uniformen Raumpartitionierung.

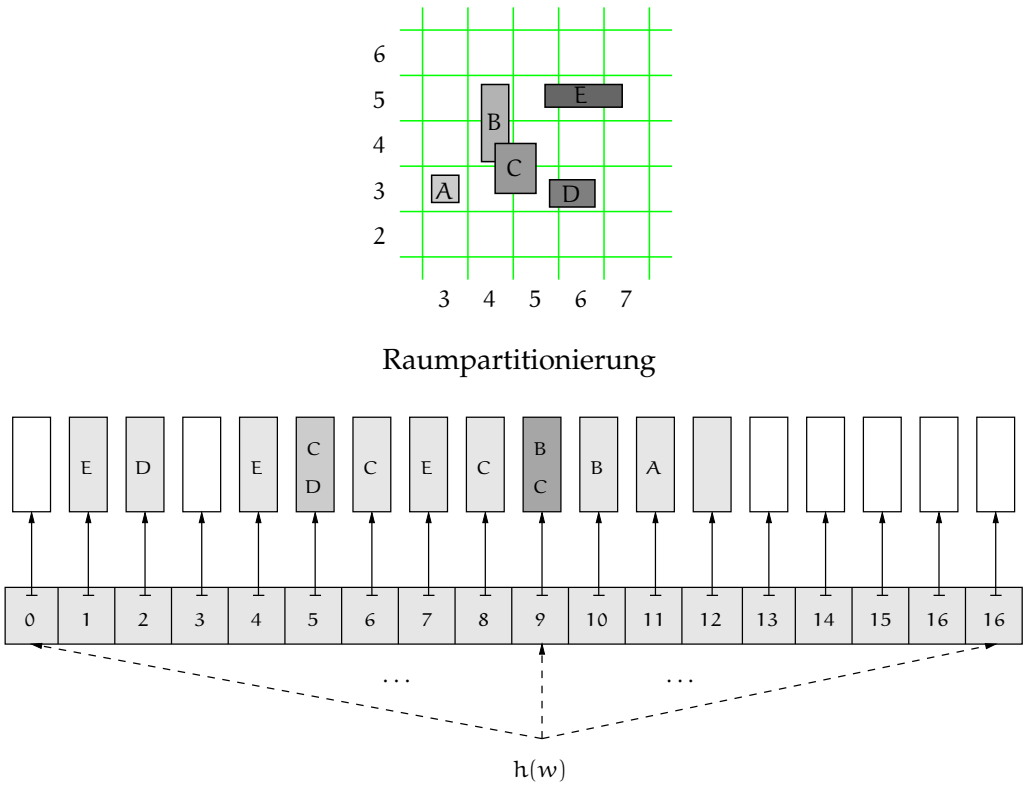
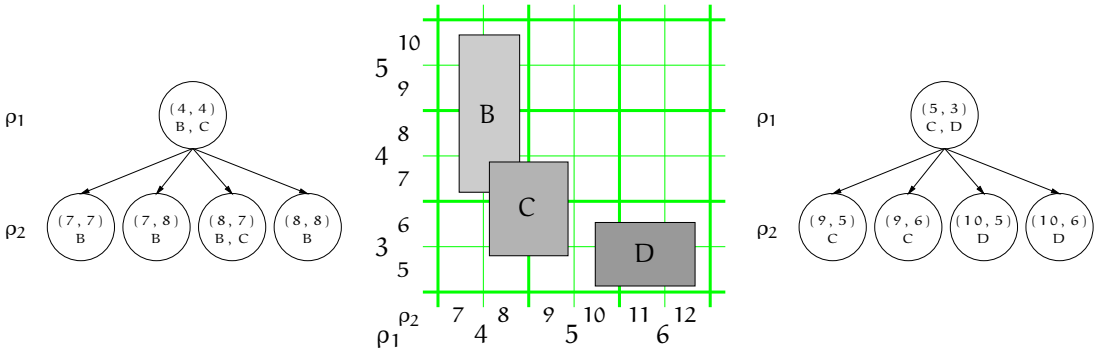


Abbildung 4.6 Beispiel für die Verwendung einer hierarchischen Raumpartitionierung zum Nachweis der räumlichen Trennung von Iso-Box C und D.



MIRTICH hat dieses Problem durch Einsatz einer hierarchischen Raumpartitionierung gelöst. „Hierarchisch“ bedeutet in diesem Kontext, daß verschiedene Raumpartitionierungen, die auf unterschiedlichen Gitterweiten beruhen, innerhalb der Datenstruktur zur Verfügung stehen. Auf diese Weise können Paare von Iso-Boxen je nach Bedarf unter verschiedenen Auflösungen des Gitters auf Schnitt getestet werden. Abbildung 4.6 veranschaulicht die Idee an unserem Beispiel aus 4.5.

Zur Umsetzung dieses einfachen Konzeptes verwendet man mehrere Hashtabellen, die die Lage der Boxen in Gittern verschiedener Weite ρ repräsentieren. Die erforderlichen Auflösungen sind offensichtlich abhängig von der Größe der zu betrachtenden Boxen. Daher definieren wir:

Definition 4.2 (Größe einer achsenorientierten Box). Sei $A = AABB(\mathbf{d}^{min}, \mathbf{d}^{max})$ eine achsenorientierte Box, dann verstehen wir unter der Größe von A, $s(A)$, den maximalen Abstand zweier gegenüberliegender Eckpunkte:

$$s(A) := \|\mathbf{d}^{max} - \mathbf{d}^{min}\| .$$

□

Jeder Box A ordnen wir nun eine Gitterweite ρ derart zu, daß das Verhältnis der Größe von A zu der Weite der Raumwürfel durch zwei Konstanten α und β, mit $0 < \alpha < 1$ und $\beta \geq 1$ nach oben und unten beschränkt ist. Wir suchen also für derart vorgegebene Konstanten α und β eine Folge von Gitterweiten $(\rho_1, \rho_2, \ldots, \rho_k)$, mit

$$\rho_1 > \rho_2 > \ldots > \rho_k > 0, \qquad k \geq 1,$$

so daß für alle Iso-Boxen A der Szene eine natürliche Zahl i_A, $1 \leq i_A \leq k$, existiert, für die gilt:

$$\alpha \leq \frac{s(A)}{\rho_{i_A}} \leq \beta .$$

Die gesuchten Gitterweiten können nun folgendermaßen ermittelt werden. Zunächst bestimmen wir die Box mit minimaler sowie maximaler Größe. Bezeichnen wir die beiden Werte mit s_{min} bzw. s_{max}, so können wir die kleinste Gitterweite festlegen:

$$\rho_1 := \frac{s_{min}}{\alpha} .$$

Somit gilt für alle Boxen A der Szene: $\alpha \leq \frac{s(A)}{\rho_1} = \frac{s(A)}{s_{min}}\alpha$. Als nächstes vergrößern wir die Gitterweite im Verhältnis β/α. Somit ergibt sich:

$$\rho_i := \frac{\beta}{\alpha}\rho_{i-1} , \qquad 2 \leq i \leq k$$

und insgesamt:

$$\rho_i = \left(\frac{\beta}{\alpha}\right)^{i-1} \rho_1 = \left(\frac{\beta}{\alpha}\right)^{i-1} \frac{s_{min}}{\alpha} .$$

Die Zahl k wird schließlich durch die Ausdehnung der größten Box definiert:

$$\begin{aligned} & \alpha \leq \frac{s_{max}}{\rho_k} \leq \beta \\ \iff \quad & \alpha \leq \frac{s_{max}}{\left(\frac{\beta}{\alpha}\right)^{k-1} \frac{s_{min}}{\alpha}} \leq \beta \\ \iff \quad & \log_{\frac{\beta}{\alpha}} \frac{s_{max}}{s_{min}} \leq k \leq \log_{\frac{\beta}{\alpha}} \frac{s_{max}}{s_{min}} + 1 . \end{aligned}$$

Da wir die Zahl der Gitterweiten möglichst gering halten wollen, wählen wir

$$k = \left\lceil \log_{\frac{\beta}{\alpha}} \frac{s_{max}}{s_{min}} \right\rceil \tag{4.2}$$

Definition 4.3 (Auflösung einer achsenorientierten Box). In einer durch α und β definierten hierarchischen Raumpartitionierung versteht man unter der Auflösung einer achsenorientierten Box, $res(A)$, die kleinste natürliche Zahl i_A, $1 \leq i_A \leq k$, für die gilt:

$$\alpha \leq \frac{s(A)}{\rho_{i_A}} \leq \beta . \tag{4.3}$$

□

Die Idee des Verfahrens von MIRTICH besteht nun darin, die Iso-Box A ausschließlich in der Hashtabelle mit Gitterweite $\rho_{res(A)}$ zu speichern, da die Auflösung dieser Tabelle in sinnvollem Verhältnis zu der Größe von A steht. Wählen wir $\alpha = \frac{1}{2}$ und $\beta = 1$, so bedeutet dies, daß alle durch das Gitter vorgegebenen Raumwürfel eine Weite $\rho_{res(A)}$ besitzen, die zwischen $s(A)$ und $2s(A)$ liegt. Da zwei achsenorientierte Boxen A_1 und A_2 verschiedene Auflösungen besitzen und somit in Hashtabellen unterschiedlicher Auflösung abgelegt sein können, muß man ihre räumliche Nähe in einer der beiden Hashtabellen überprüfen. Es macht dabei Sinn, die kleinere Auflösung, d.h. die Raumpartitionierung mit geringerer Gitterweite zu wählen.

Definition 4.4 (Überlappungskandidat). Zwei achsenorientierte Boxen A_1 und A_2 heißen Überlappungskandidat (A_1, A_2) , falls für $res^* = \min\{res(A_1), res(A_2)\}$ gilt:

$$\exists \boldsymbol{w}_1 \in W_{\rho_{res^*}}(A_1),\, \boldsymbol{w}_2 \in W_{\rho_{res^*}}(A_2) \,:\, \boldsymbol{w}_1 = \boldsymbol{w}_2$$

□

In diesem Fall ist die hierarchische Raumpartitionierung nicht in der Lage, die Schnittfreiheit der beiden Boxen zu verifizieren, so daß man den in 4.4.2 vorgestellten Überlappungstest einsetzen muß, um eine endgültige Entscheidung treffen zu können. Aus diesem Grund interessieren wir uns für die *Auflösungsschärfe* der hierarchischen Raumpartitionierung. Das folgende Lemma macht hierzu eine Aussage.

Lemma 4.2. *Zwei Iso-Boxen A_1 und A_2, mit $A_1 \cap A_2 = \emptyset$ werden von der hierarchischen Raumpartitionierung als schnittfrei identifiziert, falls für den Abstand der beiden Boxen gilt:*

$$\delta(A_1, A_2) > \frac{1}{\alpha}\sqrt{3}\max\{s(A_1), s(A_2)\}\,.$$

Beweis. O.B.d.A. sei $s(A_1) \geq s(A_2)$ und somit $res(A_1) \leq res(A_2)$. Die beiden Boxen werden als Überlappungskandidat identifiziert, falls sie unter der Auflösung $res^* = res(A_1)$ einen Raumwürfel gemeinsam besetzen. Die Seitenlänge dieses Voxels entspricht der Gitterweite $\rho_{res(A_1)}$, so daß der größte Abstand zwischen zwei Punkten dieses Würfels $\|\rho_{res(A_1)}(1,1,1)^T\| = \sqrt{3}\rho_{res(A_1)}$ beträgt. Unter Berücksichtigung von 4.3 folgt für den maximalen Abstand eines Überlappungskandidaten:

$$\delta(A_1, A_2) \leq \frac{\rho_{res(A_1)}}{s(A_1)}\sqrt{3}s(A_1) \leq \frac{1}{\alpha}\sqrt{3}s(A_1)\,.$$

□

Fügt man die Iso-Boxen der Szene in der Reihenfolge nichtfallender Auflösungen in die Datenstruktur ein, so erlauben die folgenden Tests, alle Überlappungskandidaten zu identifizieren. Bevor die Box A_1 in der Tabelle mit Auflösung $res(A_1)$ gespeichert wird, lokalisiert man A_1 in allen Raumpartitionierungen geringerer Auflösung. Befindet sich in einem Hashbucket eine Box A_2, die den gleichen Raumwürfel wie A_1 schneidet, so wird das Paar (A_1, A_2) als Überlappungskandidat gemeldet. Gehen wir im folgenden davon aus, daß unsere Tabellen auf der Technik des *perfekten Haschings* beruhen, so kann man eine Aussage über die Zahl der beim Einfügen zu betrachtenden Hashbuckets machen.

Lemma 4.3. *Die Zahl der zu untersuchenden Hashbuckets beim Einfügen einer Box in die hierarchische Hashtabelle beträgt $O\left(\beta^3 \log \frac{s_{max}}{s_{min}}\right)$ und ist somit konstant.*

Beweis. Sei A die einzufügende Box. Dann müssen wir A in allen Raumpartitionierungen mit Auflösungen i, $1 \leq i \leq res(A)$, lokalisieren. Aus $s(A) \leq \beta\rho_{res(A)}$, folgt, daß

A höchstens $(\beta+1)^3$ Raumwürfel der Seitenlänge $\rho_{res(A)}$ schneiden kann. Für Auflösungen i, $1 \leq i < res(A)$, kann diese Zahl nicht größer sein, da die Gitterweite mit abnehmender Auflösung zunimmt. Entsprechend unserer Konstruktion und Abschätzung 4.2 gilt nun:

$$res(A) \leq k = \left\lceil \log_{\frac{\beta}{\alpha}} \frac{s_{max}}{s_{min}} \right\rceil ,$$

so daß wir höchstens

$$(\beta+1)^3 \left\lceil \log_{\frac{\beta}{\alpha}} \frac{s_{max}}{s_{min}} \right\rceil = O\left(\beta^3 \log \frac{s_{max}}{s_{min}}\right) .$$

Hashbuckets betrachten müssen. □

Somit kann die Laufzeit des Verfahrens folgendermaßen abgeschätzt werden:

Satz 4.4. *Mit Hilfe der hierarchischen Hashtabelle können die Überlappungskandidaten einer Szene mit n achsenorientierten Boxen in erwarteter Zeit $O(n+c)$ bestimmt werden, wobei c der Zahl der Überlappungskandidaten entspricht.*

Beweis. Nach Lemma 4.3 betrachtet man beim Einfügen der Boxen O(n) Buckets. Für jeden Bucket melden wir die ihm zugeteilten achsenorientierten Boxen als potentiellen Überlappungspartner. Unter Verwendung von perfektem Hashing sind dies, über alle betrachteten Buckets gezählt, gerade c Boxenpaare. □

Die Überlappungskandidaten müssen anschließend mit Hilfe des exakten Tests aus 4.4.2 auf Schnitt geprüft werden. Da jeder dieser Elementartests konstante Zeit kostet, können wir folgern:

Korollar 4.5. Das vorgestellte, erweiterte Raumpartitionierungsverfahren erlaubt alle Paare von überlappenden Iso-Boxen in erwarteter Zeit $O(n+c)$ zu identifizieren.

Die optimale Wahl von α und β gestaltet sich offensichtlich schwierig, wie die gerade präsentierten Ergebnisse zeigen:

- Je größer $\frac{\beta}{\alpha}$, desto weniger Tabellen werden benötigt, d.h. desto weniger Boxenlokalisationen müssen beim Einfügen durchgeführt werden (vgl. Abschätzung 4.4.5).

- Je größer α ist, d.h. je näher α bei 1 liegt, desto besser ist die Auflösung der hierarchischen Raumpartitionierung (vgl. Lemma 4.2).

- Je kleiner β ist, d.h. je näher β bei 1 liegt, desto geringer ist die Zahl der Buckets, die beim Einfügen überprüft werden müssen (vgl. Lemma 4.3).

4.4.6 Das dynamische Kohärenzhashingverfahren

Die Vorgehensweise, wie wir sie bisher vorgestellt haben, setzt voraus, daß alle Hashtabellen vor dem Einfügen der ersten Box leer sind. Somit muß nach jedem Zeitschritt die Datenstruktur in ihren Initialisierungszustand zurückgesetzt und anschließend neu aufgebaut werden. Wenn wir jedoch Kohärenz zwischen zwei Probleminstanzen unterstellen, so ist ein großer Teil der dabei durchgeführten Einfüge- und Löschoperationen überflüssig. Liegt Kohärenz vor, so können wir annehmen, daß sich weder die Größe noch die Lage einer Box im nächsten Zeitschritt gegenüber der in der Datenstruktur gespeicherten Situation wesentlich verändert hat. Dies ist selbstverständlich insbesondere dann gegeben, wenn Boxen statische Objekte repräsentieren. Im Kohärenzfall bietet es sich daher an, lediglich die durch die aktuelle Boxenform und -lage verursachten Änderungsoperationen vorzunehmen.

Um die Datenstruktur in diesem Sinne effizient aktualisieren zu können, schlägt MIRTICH [Mir97a] die folgende Vorgehensweise vor. Man speichert den Verweis auf eine Box A nicht mehr nur in der Tabelle mit Auflösung $res(A)$, sondern zusätzlich in allen Raumpartitionierungen kleinerer Auflösung. Des weiteren führt man einen Zähler $c(A_1, A_2)$ für jedes Boxenpaar (A_1, A_2) ein, der die Zahl der von A_1 und A_2 gemeinsam besetzten Raumwürfel im Zeitablauf verfolgt. Dieser wird zu Beginn des Verfahrens mit 0 initialisiert. Da die Datenstruktur vor dem ersten Zeitschritt leer ist, können wir keine Kohärenz ausnutzen. Daher verwenden wir zum Aufbau der Datenstruktur sowie der damit einhergehenden Identifikation der Überlappungskandidaten das statische Hashingverfahren aus dem vorangegangenen Abschnitt. Der Unterschied besteht neben der gerade erwähnten Speicherung des AABB-Verweises in den Buckets niedrigerer Auflösung, in der Inkrementierung des Zählers $c(A_1, A_2)$, falls A_1 und A_2 in einem gemeinsamen Bucket der Auflösung $\min\{res(A_1), res(A_2)\}$ liegen. Bei den Iso-Boxen-Paaren (A_1, A_2) mit $c(A_1, A_2) > 0$ handelt es sich gerade um die gesuchten Überlappungskandidaten. Mit Hilfe dieses Zählers können wir in den folgenden Zeitschritten durch einfache Inkrement- und Dekrementoperationen, die Paare von Boxen bestimmen, die die Menge der Überlappungskandidaten verlassen bzw. in die Menge aufgenommen werden müssen.

Im Rahmen der Aktualisierung der Box A_1 betrachten wir sukzessive die einzelnen Hashtabellen, beginnend mit der Tabelle der Auflösung $res(A_1)$. Wird A_1 einem neuen Bucket zugeordnet, so ist der Verweis auf A_1 in diesem zu speichern. Des weiteren muß für jede, in diesem Bucket abgelegte Box A_2 der Zähler $c(A_1, A_2)$ inkrementiert werden, falls $res(A_1) = \min\{res(A_1), res(A_2)\}$ gilt. Verläßt A_1 dagegen einen Bucket des letzten Zeitschrittes, so wird der Zähler für alle Paare (A_1, A_2) mit $res(A_1) = \min\{res(A_1), res(A_2)\}$ dekrementiert, da (A_1, A_2) innerhalb des Raumwürfel als Überlappungskandidat ausgeschlossen werden kann. Falls für eine betrachtete Auflösung die von A_1 geschnittenen Raumwürfel mit denen des letzten Zeitschrittes übereinstimmen, können wir die Aktualisierung von A_1 beenden, da sich die Zuordnung zu Raumwürfeln auch in Tabellen niedrigerer Auflösung nicht geändert haben kann (Voxel haben größere Seitenlänge). Offensichtlich kann durch diese Vorgehens-

weise die folgende *Invariante* aufrechterhalten werden:

$$(A_1, A_2) \text{ ist Überlappungskandidat} \quad \Longleftrightarrow \quad c(A_1, A_2) > 0 \,.$$

Die Effizienz des Verfahren hängt somit wesentlich davon ab, wie schnell die neuhinzukommenden bzw. zu löschenden Bucketeinträge für eine Box A bei gegebener Auflösung bestimmt werden können. Unsere Aufgabe können wir somit in zwei zu lösende Teilprobleme zerlegen. Zunächst wollen wir die Verweise auf A aus den Hashbuckets löschen, die mit Raumwürfeln assoziiert sind, welche von A im nächsten Zeitschritt nicht mehr geschnitten werden. Anschließend sind Verweise auf A in allen Buckets einzufügen, die neu besetzten Voxeln entsprechen. Wir nehmen an, daß die zu aktualisierende Box A im letzten Zeitintervall die Voxel der Raumpartitionierung mit Gitterweite ρ geschnitten hat, die durch $\mathbf{o}^{\min} = \xi_\rho(\mathbf{d}^{\min})$ sowie $\mathbf{o}^{\max} = \xi_\rho(\mathbf{d}^{\max})$ definiert sind. Die resultierende Menge $W_\rho(A)$ bezeichnen wir im folgenden mit O. Es gilt daher:

$$O := \left\{ \boldsymbol{W} \in \mathbb{Z}^3 \,\middle|\, \forall i, 1 \le i \le 3 : \; o_i^{\min} \le w_i \le o_i^{\max} \right\} .$$

Für das nächste Zeitintervall wird die Box bei gleicher Auflösung folgendermaßen lokalisiert:

$$N := \left\{ \boldsymbol{W} \in \mathbb{Z}^3 \,\middle|\, \forall i, 1 \le i \le 3 : \; n_i^{\min} \le w_i \le n_i^{\max} \right\} .$$

Die Vektoren $\mathbf{o}^{\min}$, $\mathbf{o}^{\max}$ sowie $\mathbf{n}^{\min}$ und $\mathbf{n}^{\max}$ charakterisieren somit die Lage der Box bezüglich einer Raumpartitionierung fester Auflösung in zwei aufeinanderfolgenden Zeitschritten. Wir interessieren uns im Rahmen unserer Aufgabenstellung für zwei Mengen von Raumwürfeln. Dabei handelt es sich zum einen um die Menge der Voxel D aus O, die von A im nächsten Zeitschritt nicht mehr geschnitten werden:

$$D := O \setminus (O \cap N) = O \setminus K \,,$$

mit $K = O \cap N = \left\{ \boldsymbol{W} \in \mathbb{Z}^3 \,\middle|\, \forall i, 1 \le i \le 3 : \; k_i^{\min} \le w_i \le k_i^{\max} \right\}$.
Zum anderen müssen wir die Menge der Raumwürfel I bestimmen, die im nächsten Zeitschritt besetzt werden, bisher jedoch nicht geschnitten wurden:

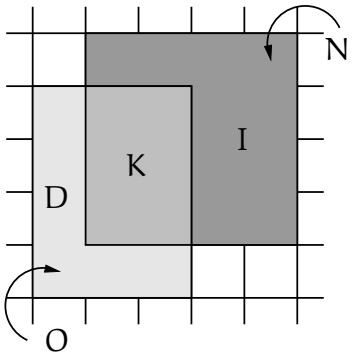
$$I := N \setminus K \,.$$

Abbildung 4.7 verdeutlicht unsere Aufgabe im zweidimensionalen Fall. Die zu bestimmenden Mengen D und I sind in der Graphik schattiert gezeichnet. Wir wollen zunächst die Menge K als dreidimensionales Intervall angeben:

$$\begin{aligned} K &= [k_1^{\min}, k_1^{\max}] \times [k_2^{\min}, k_2^{\max}] \times [k_3^{\min}, k_3^{\max}] \\ &= \left[\max\{o_1^{\min}, n_1^{\min}\}, \min\{o_1^{\max}, n_1^{\max}\} \right] \times \left[\max\{o_2^{\min}, n_2^{\min}\}, \min\{o_2^{\max}, n_2^{\max}\} \right] \times \\ &\quad \left[\max\{o_3^{\min}, n_3^{\min}\}, \min\{o_3^{\max}, n_3^{\max}\} \right] . \end{aligned}$$

Die Menge können wir nun als Vereinigung folgender Teilmengen D_{ij}, $1 \le i, j \le 3$,

Abbildung 4.7 Die drei Mengen D, I und K, die durch Schnitt zweier quaderförmiger Raumwürfelmengen O und N entstehen.



darstellen:

$$
\begin{aligned}
D_{11} &= \left[o_1^{min},\, \min\{o_1^{max}, n_1^{min}-1\}\right] \times \left[o_2^{min}, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
&= \left[o_1^{min},\, \min\{o_1^{max}, k_1^{min}-1\}\right] \times \left[o_2^{min}, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
D_{12} &= \left[\max\{o_1^{min}, n_1^{max}+1\},\, o_1^{max}\right] \times \left[o_2^{min}, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
&= \left[\max\{o_1^{min}, k_1^{max}+1\},\, o_1^{max}\right] \times \left[o_2^{min}, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
D_{21} &= \left[k_1^{min}, k_1^{max}\right] \times \left[o_2^{min},\, \min\{o_2^{max}, n_2^{min}-1\}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
&= \left[k_1^{min}, k_1^{max}\right] \times \left[o_2^{min},\, \min\{o_2^{max}, k_2^{min}-1\}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
D_{22} &= \left[k_1^{min}, k_1^{max}\right] \times \left[\max\{o_2^{min}, n_2^{max}+1\},\, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
&= \left[k_1^{min}, k_1^{max}\right] \times \left[\max\{o_2^{min}, k_2^{max}+1\},\, o_2^{max}\right] \times \left[o_3^{min}, o_3^{max}\right] \\
D_{31} &= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[o_3^{min},\, \min\{o_3^{max}, n_3^{min}-1\}\right] \\
&= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[o_3^{min},\, \min\{o_3^{max}, k_3^{min}-1\}\right] \\
D_{32} &= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[\max\{o_3^{min}, n_3^{max}+1\},\, o_3^{max}\right] \\
&= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[\max\{o_3^{min}, k_3^{max}+1\},\, o_3^{max}\right].
\end{aligned}
$$

Es gilt nun:

$$D = \bigcup_{1 \le i,j \le 3} D_{ij}\,.$$

Die Menge $I = N \setminus K$ ergibt sich nun analog als

$$I = \bigcup_{1 \le i,j \le 3} I_{ij}\,,$$

mit

$$
\begin{aligned}
I_{11} &= \left[n_1^{min},\, \min\{n_1^{max}, k_1^{min}-1\}\right] \times \left[n_2^{min}, n_2^{max}\right] \times \left[n_3^{min}, n_3^{max}\right] \\
I_{12} &= \left[\max\{n_1^{min}, k_1^{max}+1\},\, n_1^{max}\right] \times \left[n_2^{min}, n_2^{max}\right] \times \left[n_3^{min}, n_3^{max}\right] \\
I_{21} &= \left[k_1^{min}, k_1^{max}\right] \times \left[n_2^{min},\, \min\{n_2^{max}, k_2^{min}-1\}\right] \times \left[n_3^{min}, n_3^{max}\right] \\
I_{22} &= \left[k_1^{min}, k_1^{max}\right] \times \left[\max\{n_2^{min}, k_2^{max}+1\},\, n_2^{max}\right] \times \left[n_3^{min}, n_3^{max}\right] \\
I_{31} &= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[n_3^{min},\, \min\{n_3^{max}, k_3^{min}-1\}\right] \\
I_{32} &= \left[k_1^{min}, k_1^{max}\right] \times \left[k_2^{min}, k_2^{max}\right] \times \left[\max\{n_3^{min}, k_3^{max}+1\},\, n_3^{max}\right].
\end{aligned}
$$

Algorithmus 29 berechnet mit Hilfe der soeben formulierten dreidimensionalen Intervalle die Raumwürfel innerhalb der Raumpartitionierung vorgegebener Auflösung, in denen ein Verweis auf A gelöscht bzw. eingefügt werden muß.

Algorithmus 29 Ein Algorithmus zur Bestimmung der Mengen D und I.

Eingabe: Die Vektoren $\mathbf{o}^{min}$, $\mathbf{o}^{max}$, $\mathbf{n}^{min}$ und $\mathbf{n}^{max}$, die die Mengen O und N der in zwei aufeinanderfolgenden Zeitschritten besetzten Raumwürfel beschreiben.
Ausgabe: Die Mengen $D = O \setminus (O \cap N)$ und $I = N \setminus (O \cap N)$.

VOXELUPDATESETS$((\mathbf{o}^{min}, \mathbf{o}^{max}), (\mathbf{n}^{min}, \mathbf{n}^{max}))$
(1) $D \leftarrow \emptyset$
(2) $I \leftarrow \emptyset$
(3) **for** $i \leftarrow 1$ **to** 3
(4) $\quad k_i^{min} \leftarrow \max\{o_i^{min}, n_i^{min}\}$
(5) $\quad k_i^{max} \leftarrow \min\{o_i^{max}, n_i^{max}\}$
(6) $\quad D \leftarrow D_{i1} \cup D_{i2}$
(7) $\quad I \leftarrow I_{i1} \cup I_{i2}$
(8) $\quad$ **if** $k_i^{min} > k_i^{max}$
(9) $\quad\quad$ **return** $[D, I]$
(10) **return** $[D, I]$

Das dynamische Kohärenzhashingverfahren nach MIRTICH verzichtet auf die Überprüfung der Überlappungskandidaten mit Hilfe eines exakten Schnittests. Es betrachtet daher alle Paare von Bewegungshüllkörpern als nahe, die im zugrundeliegenden Zeitintervall einen gemeinsamen Raumwürfel besetzen. Dies liefert den Algorithmus CLOSEPAIRS (Algorithmus 30), der die Menge der Körperpaare bestimmt, die sich im zu betrachtenden Zeitintervall nahe kommen und somit in den Kollisionsheap aufgenommen werden müssen. Mit T_{ρ_i} bezeichnen wir die Hashtabelle, die zur Boxenlokalisation innerhalb einer Raumpartitionierung der Gitterweite ρ_i, $1 \leq i \leq k$ herangezogen wird. $\tilde{A}_1 = AABB(\tilde{d}_1^{min}, \tilde{d}_1^{max})$ sowie $A = AABB(d_1^{min}, d_1^{max})$ seien die Bewegungshüllkörper von $\mathcal{K}_1$ im letzten bzw. im nächsten Zeitschritt.

Algorithmus 30 Ein Algorithmus zur Bestimmung der Körperpaare, die in den Heap aufgenommen werden müssen.

Eingabe: Die Welt $\mathcal{W}$ sowie das zu simulierende Zeitintervall $[t_0, t_1]$.
Ausgabe: Die Menge S der Körperpaare, die sich im Zeitintervall $[t_0, t_1]$ nahe kommen, bisher jedoch als entfernt klassifiziert wurden.

CLOSEPAIRS($\mathcal{W}, t_0, t_1$)
(1) **foreach** $\mathcal{K}_1 \in \mathcal{W}$
(2) $\tilde{A}_1 \leftarrow A_1$
(3) $A_1 \leftarrow AABB(\mathcal{K}_1, t_0, t_1)$
(4) **for** $i \leftarrow res(A_1)$ **downto** 1
(5) $(\mathbf{o}^{min}, \mathbf{o}^{max}) \leftarrow \left(\xi_{\rho_i}(\tilde{\mathbf{d}}^{min}), \xi_{\rho_i}(\tilde{\mathbf{d}}^{max})\right)$
(6) $(\mathbf{n}^{min}, \mathbf{n}^{max}) \leftarrow \left(\xi_{\rho_i}(\mathbf{d}^{min}), \xi_{\rho_i}(\mathbf{d}^{max})\right)$
(7) $[D, I] \leftarrow$ VOXELUPDATESETS$((\mathbf{o}^{min}, \mathbf{o}^{max}), (\mathbf{n}^{min}, \mathbf{n}^{max}))$
(8) **if** $D = \emptyset \wedge I = \emptyset$
(9) **break**
(10) **foreach** $\boldsymbol{W} \in D$
(11) $T_{\rho_i}[\boldsymbol{W}] \leftarrow T_{\rho_i}[\boldsymbol{W}] \setminus \{A_1\}$
(12) **foreach** $A_2 \in T_{\rho_i}[\boldsymbol{W}]$
(13) **if** $res(A_1) \leq res(A_2)$
(14) $c(A_1, A_2) \leftarrow c(A_1, A_2) - 1$
(15) **if** $(\mathcal{K}_1, \mathcal{K}_2) \in S$
(16) $S \leftarrow S \setminus \{(\mathcal{K}_1, \mathcal{K}_2)\}$
(17) **foreach** $\boldsymbol{W} \in I$
(18) **foreach** $A_2 \in T_{\rho_i}[\boldsymbol{W}]$
(19) **if** $res(A_1) \leq res(A_2)$
(20) $c(A_1, A_2) \leftarrow c(A_1, A_2) + 1$
(21) **if** $c(A_1, A_2) = 1$
(22) $S \leftarrow S \cup \{(\mathcal{K}_1, \mathcal{K}_2)\}$
(23) $T_{\rho_i}[\boldsymbol{W}] \leftarrow T_{\rho_i}[\boldsymbol{W}] \cup \{A_1\}$
(24) **return** S

4.5 Die Simulationsschleife

Aus den bisherigen Überlegungen ist offensichtlich geworden, daß der vorgestellte Kollisionserkennungsansatz den Ablauf der Simulationsschleife steuert. Die Abtastung der Distanzfunktion liefert Zeitintervalle, in denen sich die zu simulierenden Körper auf ballistischen Bewegungsbahnen und somit kollisionsfrei bewegen. Dies ist die Voraussetzung, um das dynamische System durch Integration der Bewegungsgleichungen zu simulieren. Jeder Schritt der Kollisionserkennung entspricht somit einem Simulationsschritt, was wir bereits in Abschnitt 4.3.1 angedeutet haben. Dennoch wollen wir an dieser Stelle das Konzept der Simulationssteuerung, das Bewegungsintegrati-

on, Kollisionserkennung sowie Kollisionsauflösung in einen zeitlichen und logischen Zusammenhang stellt, formalisieren. Algorithmus 31 skizziert den Ablauf der Simulationsschleife. Dabei wird die Routine COLLISIONRESPONSE verwendet, die im Fall einer Kollision zweier Körper $\mathcal{K}_1$, $\mathcal{K}_2$ die Kollisionsimpulse berechnet, die eine Durchdringung der Kollisionspartner verhindern. Diese Impulse bewirken eine unmittelbare Geschwindigkeitsänderung von $\mathcal{K}_1$ und $\mathcal{K}_2$, so daß sich beide Körper nach Auflösung der Kollision wieder auf ballistischen Bewegungsbahnen befinden. Die Prozedur INTEGRATEDYNAMICSYSTEM führt schließlich den eigentlichen Simulationsschritt durch Integration der Bewegungsgleichungen aller Körper durch. In jedem Simulationsschritt wird das Körperpaar, das den frühsten möglichen Kollisionszeitpunkt definiert, aus dem Heap entfernt und einem Abstandstest unterzogen. Liegt eine Kollision vor, da die ermittelte Distanz den Toleranzwert ε_c unterschritten hat, so müssen wir die Kollision auflösen und anschließend die Schlüssel der betroffenen Heapelemente aktualisieren (vgl. 4.3.2 und 4.3.3). Liegt dagegen keine Kollision vor, so ist die Kollisionszeitschranke von $(\mathcal{K}_1, \mathcal{K}_2)$ zu korrigieren und das Körperpaar unter dem aktualisierten Schlüssel in den Heap einzufügen. Dabei ist jedoch das Hysteresis-Konzept aus 4.3.5 zu beachten, welches sicherstellt, daß nur solche Paare wieder in den Heap aufgenommen werden, die sich im letzten Zeitschritt nahe waren. Um die Länge des nächsten Simulationsschrittes zu bestimmen, wird wie in 4.3.5 erläutert, zunächst der frühste Kollisionszeitpunkt t_L aller Heapelemente betrachtet. Anschließend werden alle Körperpaare, die sich im Zeitintervall $[t_0, t_L]$ nahe kommen, jedoch noch nicht im Heap repräsentiert sind, in diesen aufgenommen. Der frühste Kollisionszeitpunkt aller im Intervall $[t_0, t_L]$ nahen Körperpaare definiert das Ende t'_L des nächsten Simulationsschrittes. Dieser wird schließlich durch Integration der Bewegungsgleichungen durchgeführt.

4.6 Zusammenfassung

Abbildung 4.8 veranschaulicht den Aufbau des Kollisionserkennungssystems sowie die Kommunikation zwischen den Teilkomponenten. Um die Steuerungsfunktion der Kollisionserkennung im Gesamtsystem zu verdeutlichen, ist zusätzlich das Zusammenspiel mit den Modulen zur Kollisionsauflösung sowie zur Ermittlung und Entwicklung des Dynamikzustandes dargestellt. Entsprechend unseren Überlegungen in Abschnitt 2.6.3 unterscheiden wir zwischen der *narrow-phase-Kollisionserkennung* und der *broad-phase-Komponente*. Zur narrow-phase-Kollisionserkennung setzen wir das in Kapitel 3 vorgestellte *Abstandsberechnungsverfahren* ein, das den Euklidischen Abstand zweier Körper sowie ein nahestes Punktepaar als Zeugen des Abstandsminimums an die broad-phase-Komponente weiterleitet. Die Distanzinformation erlaubt neben der Beantwortung der statischen Kollisionserkennungsfrage die Bestimmung des *frühsten Kollisionszeitpunktes* (KZP) zweier Körper, auf dessen Basis der Kollisionsheap (Abschnitt 4.3) aufgebaut wird. Um nicht alle Körperpaare der Simulation in der Datenstruktur verwalten zu müssen, verwenden wir die *hashingbasierte hierarchische Raumpartitionierungstechnik* aus Abschnitt 4.4.6. Durch Lokalisation der *Bewegungshüllkörper* im

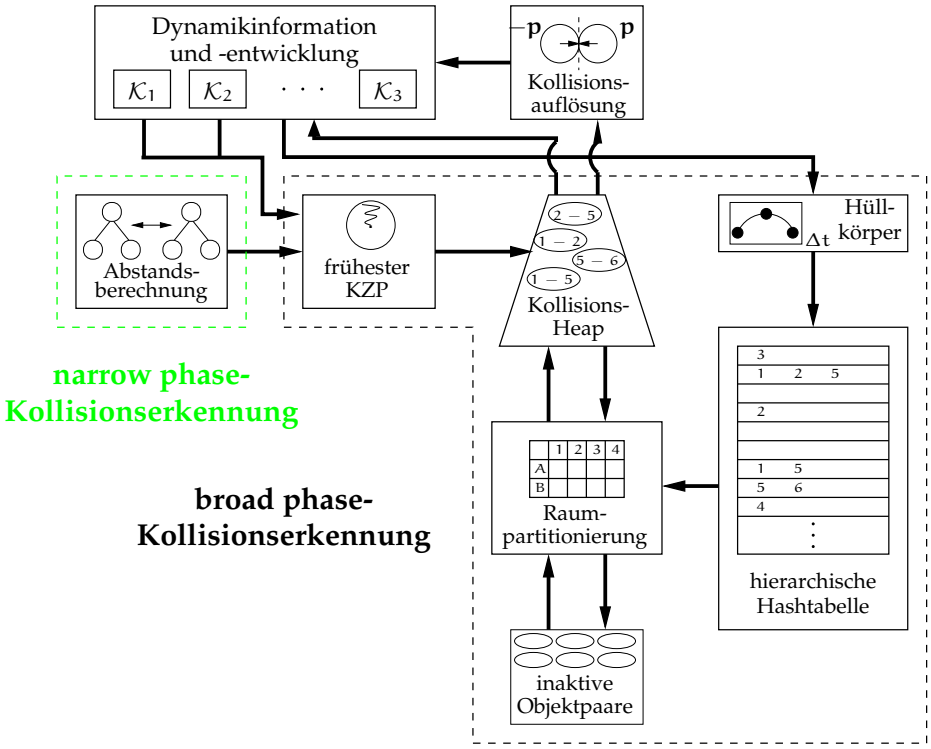
Algorithmus 31 Die Steuerung des Simulationsablaufs durch die Kollisionserkennung.

Eingabe: Die Welt $\mathcal{W}$ sowie das zu simulierende Zeitintervall $[t_0, t_1]$.

```
SIMULATE(W, t_0, t_1)
(1)   repeat
(2)     [t_L, (K_1, K_2)] ← H.DELMIN()
(3)     d ← δ(K_1, K_2)
(4)     if d < ε_c
(5)       COLLISIONRESPONSE(K_1, K_2)
(6)       UPDATEHEAP((K_1, K_2), t_0, t_1)
(7)     else if c(A_1, A_2) > 0
(8)       t_L ← TOI(d, (K_1, K_2), t_0, t_1)
(9)       H.INSERT(t_L, (K_1, K_2))
(10)    [t_L, (K_1, K_2)] ← H.DELMIN()
(11)    S ← CLOSEPAIRS(W, t_0, t_L)
(12)    foreach (K_1, K_2) ∈ S
(13)      d ← δ(K_1, K_2)
(14)      t ← t_0 + TOI(d, (K_1, K_2), t_0, t_1)
(15)      H.INSERT(t, (K_1, K_2))
(16)    t'_L ← min{H.MINKEY(), t_1}
(17)    INTEGRATEDYNAMICSYSTEM(W, t_0, t'_L)
(18)    t_0 ← t'_L
(19)  until t_0 = t_1
```

Raum können alle die Objektpaare identifiziert werden, die sich im nächsten zu simulierenden Zeitschritt nahe kommen. Die Simulation einer ballistischen Bewegungsphase, welche durch das Kollisionserkennungssystem ausgelöst und deren Länge durch das oberste Element des Kollisionsheaps definiert wird, ist Aufgabe des Moduls zur Entwicklung des Dynamikzustandes (Abschnitt 2.5). Diese Komponente liefert zudem alle Kinematikdaten, die zur Abstandsberechnung, zur Bestimmung des frühsten Kollisionszeitpunktes sowie im Rahmen der Konstruktion von Bewegungshüllkörpern benötigt werden. Das Kollisionserkennungssystem initiiert und terminiert jedoch nicht nur die Berechnung der Bewegungsbahnen, sondern löst selbstverständlich auch die Kollisionsauflösung aus. Diese modifiziert den Dynamikzustand des zu simulierenden Systems durch unmittelbare Änderung von linearer Geschwindigkeit und Winkelgeschwindigkeit,

Abbildung 4.8 Aufbau des Kollisionserkennungsystems.



Die Kollisionsauflösung

5 Die impulsbasierte Kollisionsauflösung

Die Überlegungen aus Abschnitt 2.5 erlauben es, die Bewegung isolierter, starrer Körper unter dem Einfluß externer Kräfte wie Gravitation zu simulieren. Der Übergang zum Mehrkörperproblem wirft jedoch neue Fragen auf, die sich aus den Interaktionsmöglichkeiten der Objekte des Systems ergeben. Das physikalische Modell, welches unserer Betrachtung zugrunde liegt, schließt Anziehungskräfte aus, so daß eine Interaktion zwischen den beteiligten Körpern ausschließlich in Form von Kollisionen stattfinden kann. Um das Verhalten des Systems vorauszusagen, ist es daher unbedingt erforderlich, die Dynamik von Kollisionen zu untersuchen und ein Modell zu entwickeln, welches den Einfluß der Kollision auf die Bewegungszustände der beteiligten Objekte beschreibt. Die Komplexität der Kontaktdynamik erzwingt dabei gewisse vereinfachende Annahmen, so daß ein Trade-Off zwischen Realitätsnähe und Echtzeitfähigkeit der Simulation aufbricht. In Abschnitt 2.2 haben wir verschiedene Modelle und Simulationsparadigmen kennengelernt, die sich hinsichtlich dieser Kriterien positionieren lassen. Die impulsbasierte Methode, welche auf Arbeiten von KELLER [Kel86] HAHN [Hah88] und STRONGE [Str90, Str91] beruht und von MIRTICH [MC95, Mir95, Mir96b] zusammengeführt und algorithmisch effizient aufgearbeitet wurde, haben wir zum Gegenstand dieses Kapitels gemacht. Der große Anreiz, diese Simulationsmethode zu verwenden, ergibt sich aus der Breite des Anwendungsspektrum, welches durch sie abgedeckt werden kann. Der Ansatz wurde daher im Rahmen des SILVIA-Projektes implementiert und bezüglich Plausibilität sowie Eignung in verschiedenen Szenarien studiert.

5.1 Das Kontaktmodell

Das Kontaktmodell, welches der impulsbasierten Simulation zugrunde liegt, basiert auf folgenden Annahmen:

1. Infinitesimal kleine Kollisionszeit,
2. Strongesche Hypothese der Energieumwandlung,
3. Coulombsches Reibungsgesetz.

5.1.1 Infinitesimal kleine Kollisionszeit

Diese Annahme ist die modelltheoretische Rechtfertigung der Ermittlung von Impulsen statt Kontaktkräften als Reaktion auf die Kollision zweier starrer Körper. Betrach-

tet man nämlich ein infinitesimal kleines Zeitintervall, dann haben alle auftretenden Kontaktkräfte Impulscharakter, während zeitunabhängige Kräfte wie z.B. Gravitation ignoriert werden können. Des weiteren gestattet uns diese Annahme, die Position und Orientierung der Kontaktpartner während der Kollision als konstant anzunehmen.

5.1.2 Strongesche Hypothese der Energieumwandlung

Während der Kollision findet eine Energieumwandlung statt. Die kinetische Energie der Kollisionspartner wird in der sogenannten *Kompressionsphase* durch Anregung innerer Freiheitsgrade in den Körpern gespeichert. Nachdem der *Zeitpunkt maximaler Kompression* erreicht ist, wird diese gespeicherte Energie in einem gewissen Umfang wieder in kinetische Energie der Kontaktpartner umgewandelt. Ob nun die gesamte aufgenommene Energie oder nur ein Teil davon in der sogenannten *Rückstellungsphase* an die Kontaktpartner zurückgegeben wird, hängt von der Art des Stoßes ab. Man spricht von *elastischen Stößen*, falls die gesamte kinetische Energie, die in innere Energie der Stoßpartner umgewandelt wurde, am Ende des Stoßes wieder auf beide Körper verteilt wird. Andernfalls handelt es sich um einen *inelastischen Stoß*, bei dem ein Teil der kinetischen Energie beispielsweise in Form von Wärme abgegeben wird. Ein wichtiger Spezialfall stellt der sogenannte *plastische Stoß* dar, bei dem die gesamte während der Kompressionsphase gespeicherte Energie den Kollisionspartnern nicht mehr zurückgegeben wird.

Da wir ausschließlich starre Körper betrachten, die über keine inneren Freiheitsgrade verfügen, führt ein inelastischer Stoß zwangsläufig zu einem Energieverlust des modellierten Systems. Somit gilt die Invariante, daß die Gesamtenergie der Stoßpartner nach der Kollision nicht größer sein kann, als sie es zu Beginn des Stoßes war.

Um den Grad der Umwandlung von innerer in kinetischer Energie berücksichtigen zu können, führt man eine Konstante e, den sogenannten *Rückstellungskoeffizienten* ein, dessen Wert im Intervall $[0, 1]$ liegt und abhängig von den Materialeigenschaften der Stoßpartner zu wählen ist.

Die *Strongesche Hypothese* ist eine einfache Regel, die die Energieumwandlung während der Kollision beschreibt.

Vermutung 1 *(*STRONGE*).* Sei W_z die Arbeit, die die Kollisionsimpulse während der Kollision in Kontaktrichtung verrichten. Dann gilt:

$$W_z(t_f) = (1 - e^2)W_z(t_{mc}), \tag{5.1}$$

wobei t_f der Endzeitpunkt der Kollision und t_{mc} der Zeitpunkt maximaler Kompression ist.

Die Bedeutung der Hypothese von STRONGE liegt darin, daß sie das einzige Modell in diesem Zusammenhang ist, für das man zeigen kann, daß die oben beschriebene Energieinvariante gilt.

5.1.3 Coulombsches Reibungsgesetz

Jeder bewegter Körper wird durch Wechselwirkung mit seiner Umgebung gebremst. Es treten bei der Bewegung Reibungskräfte auf, die seiner Bewegung entgegengerichtet sind. Der Einsatz von Reibungsmodellen ist daher unerläßlich, wenn man das Kontaktverhalten konkreter Objekte realtitätsgetreu simulieren will. Das Beispiel des Steh-Auf-Kreisels in Abschnitt 6.2.1 oder des keltischen Wackelsteines aus 6.2.4 zeigt, daß bestimmte physikalische Phänomene ohne Berücksichtigung von Kontaktreibung nicht zu erklären sind. Betrachtet man Reibung zwischen Festkörpern, so unterscheidet man im allgemeinen zwischen *Gleit-*, *Haft-* und *Rollreibung*. Daher liegt der impulsbasierten Methode das empirische *Reibungsmodell von* COULOMB zugrunde, welches im Fall von Gleit- und Haftreibung Aussagen über die wirkenden Reibungskräfte trifft.

Vermutung 2 *(Coulombsches Reibungsgesetz).* Sei $\mathbf{u}$ die relative Kontaktpunktgeschwindigkeit von Körper 1 zu Körper 2. Desweiteren bezeichnen wir mit $\mathbf{u_t}$ die Tangentialkomponente von $\mathbf{u}$ und mit $\hat{\mathbf{u}}_t$ den Einheitsvektor in diese Richtung. $\mathbf{F}_n$ und $\mathbf{F}_t$ seien die Normalen- bzw. Tangentialkomponenten der Kraft, die von Körper 2 auf Körper 1 ausgeübt wird. Dann gilt:

$$\mathbf{u_t} \neq 0 \implies \mathbf{F}_t = -\mu\|\mathbf{F}_n\|\hat{\mathbf{u}}_t\,, \tag{5.2}$$

$$\mathbf{u_t} = 0 \implies \mathbf{F}_t \leq \mu\|\mathbf{F}_n\|\,, \tag{5.3}$$

wobei $\mu \in [0, 1]$ den Reibungskoeffizienten bezeichnet.

Gleichung 5.2 beschreibt den Gleitreibungsfall (*kinetische Reibung*). Die Reibungskraft ist demnach proportional zur Größe der Kraft in Normalenrichtung, jedoch der Gleitrichtung entgegengerichtet. Verschwindet die Relativgeschwindigkeit (*statische Reibung*), wie in 5.3 beschrieben, dann tritt Haftreibung auf, welche die Tangentialkomponente der äußeren Kraft kompensiert. Dies gilt natürlich nur solange die Zugkraft eine gewisse obere Grenze nicht überschreitet. Der Leser wird an dieser Stelle sicher festgestellt haben, daß Rollreibung in dem Modell von COULOMB nicht betrachtet wird und somit von unserer Simulation nicht berücksichtigt werden kann. Da die Literatur für dieses Problem ausschließlich ad-hoc-Lösungen präsentiert, sehen wir in der Entwicklung und Integration eines Rollreibungsmodells eine sinnvolle Erweiterungsmöglichkeit der impulsbasierten Simulationsmethode.

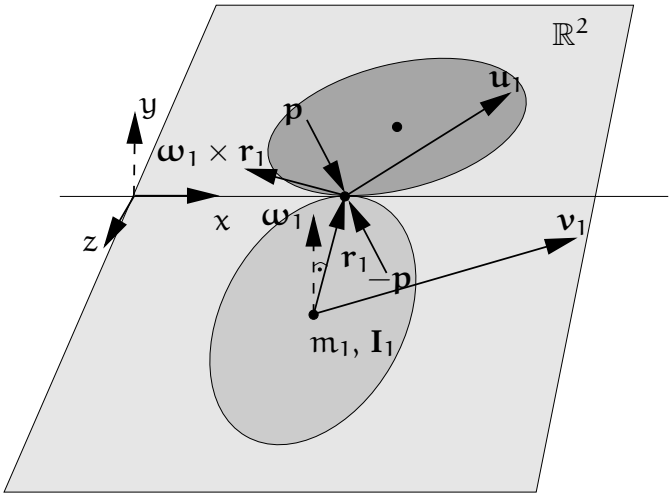
5.2 Analyserahmen und Kollisionsidentifikation

Es wird sich im folgenden als nützlich erweisen, die Kollisionsanalye in einem speziellen Koordinatensystem, dem sogenannten *Kollisionskoordinatensystem* durchzuführen. Das Kollisionskoordinatensystem ist dabei folgendermaßen definiert:

(i) Der Ursprung liegt im Kontaktpunkt.

(ii) Die z-Achse ist parallel zu dem Normalenvektor der Fläche, falls es sich um einen Punkt-Fläche-Kontakt handelt bzw. steht im Fall eines Kante-Kante-Kontaktes senkrecht auf den Richtungsvektoren der beteiligten Kanten.

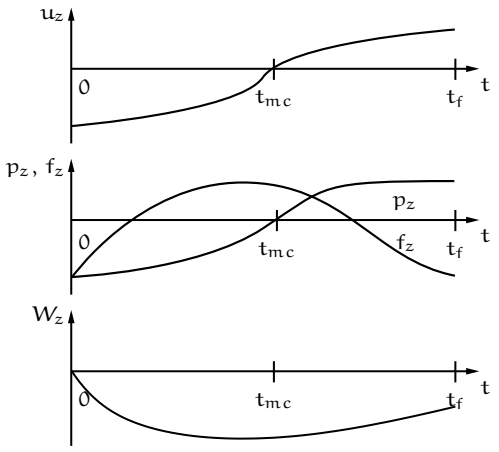
Abbildung 5.1 Die Kollision zweier Körper: Kollisionskoordinatensystem sowie analyserelevante Dynamikgrößen.



Die Wahl des Koordinatensystems definiert eine *Kontaktebene*, die von der x- und y-Achse des Kollisionskoordinatensystems aufgespannt wird. In ihr können wir die Reibungskräfte auf einfache Weise analysieren. Die *Kollisionsrichtung* entspricht der z-Achse, so daß Kontaktkräfte stets in negativer z-Richtung wirken. Die Ausgangssituation der Kollisionsanalyse stellt sich somit wie in Abbildung 5.1 beschrieben dar. Für Körper $\mathcal{K}_i$, $i = 1, 2$, sei m_i die Masse und $\mathbf{I}_i$ die Trägheitsmatrix, $\boldsymbol{\nu}_i$ die lineare bzw. $\boldsymbol{\omega}_i$ die Winkelgeschwindigkeit. Der Vektor $\mathbf{u}_i = \boldsymbol{\nu}_i + \boldsymbol{\omega}_i \times \mathbf{r}_i$ gibt die absolute Geschwindigkeit von Körper $\mathcal{K}_i$, $i = 1, 2$, im Kontaktpunkt an. Dabei bezeichnet r_i den Vektor vom Massenschwerpunkt zum Kontaktpunkt. Das Resultat der Impulsberechnung ist ein Vektor $\mathbf{P}$, der den Impuls beschreibt, welchen Körper 2 über den Stoß auf Körper 1 ausübt. Aufgrund der Impulserhaltung gilt ferner, daß $-\mathbf{P}$ als Impuls auf Körper 2 übertragen wird. Wir nehmen im folgenden an, daß alle betrachteten Größen bezüglich dieses Kollisionskoordinatensystems ausgedrückt sind. Abbildung 5.2 gibt den charakteristischen zeitlichen Verlauf der relativen Kontaktpunktgeschwindigkeit $\mathbf{u}$, des Impuls $\mathbf{p}$ sowie der Kraft $\mathbf{F}$ im **Kontaktpunkt** wieder . Desweiteren ist die Entwicklung der verrichteter Arbeit W in der Kontaktrichtung z dargestellt.

Während eine Kollision in unserer physikalischen Betrachtungsweise keine zeitliche Ausdehnung hat, dürfen wir nicht übersehen, daß die primär auf geometrischen Daten operierende Kollisionserkennung mit Toleranzen arbeitet und einen Kontakt erst dann als beendet ansieht, wenn ein Körper die ε_c-Hülle des Kontaktpartners verlassen hat. Dies bedeutet, daß die Kollsionserkennung nicht in der Lage ist, bereits aufgelöste Kollisionen von zu behandelnden Kontakten zu unterscheiden, solange die Körper sich nicht ausreichend weit voneinander entfernt haben. Daher müssen wir unserem bisherigen, rein geometrischen Kollisionsbegriff eine kinematische Forderung hinzufügen.

Abbildung 5.2 Der charakteristische Verlauf der zentralen Dynamikgrößen während der Kollision.



Definition 5.1 (Kollision zweier Objekte). Seien $\mathcal{K}_1, \mathcal{K}_2 \in \mathcal{W}$ zwei Körper unserer Simulation. $\mathcal{K}_1$ und $\mathcal{K}_2$ kollidieren genau dann, wenn folgende Kriterien erfüllt sind:

(i) $\delta(\mathcal{K}_1, \mathcal{K}_2) \leq \varepsilon_c$ **und**

(ii) $\dot{\delta}(\mathcal{K}_1, \mathcal{K}_2) < 0$.

□

Wir wollen also nur solche geometrischen Kontakte durch Kollisionsimpulse auflösen, für die gilt, daß der Abstand zwischen den nahesten Punkte abnimmt.

Beobachtung 9. Sei (q_1, q_2) das naheste Punktepaar, c_i der Massenschwerpunkt und $\mathbf{r}_i = \mathbf{q}_i - \mathbf{c}_i$ der Vektor vom Massenschwerpunkt zum nahesten Punkt q_i von Körper $\mathcal{K}_i$, $i = 1, 2$. Dann liegt eine Kollision vor, wenn

(i) die Abstandsberechnung einen Kontakt meldet **und**

(ii) die Relativgeschwindigkeit der nahesten Punkte negativ ist, d.h. wenn

$$u_z = \left[\boldsymbol{\nu}_1 + \boldsymbol{\omega}_1 \times \mathbf{r}_1 - \boldsymbol{\nu}_2 - \boldsymbol{\omega}_2 \times \mathbf{r}_2\right]_z < 0 \tag{5.4}$$

gilt.

5.3 Die Kollisionsgleichungen

In Abschnitt 2.5 haben wir beschrieben, wie sich starre Körper unter dem Einfluß von Kräften verhalten. Die modelltheoretischen Annahmen der impulsbasierten Simulationsmethode gestatten es, die Kollisionsanalyse auf lediglich zwei Kräfte zu beschränken, nämlich die Kontaktkraft $\mathbf{F}_n$ sowie die Reibungskraft $\mathbf{F}_t$, die als Gesamtkraft $\mathbf{F}$ im

Kontaktpunkt angreifen und somit ein Drehmoment $\mathbf{D}$ induzieren. Ausgehend von unseren allgemeinen Überlegungen in 2.5 wissen wir, daß Körper 1 auf Körper 2 die folgende Gesamtkraft und das folgende Drehmoment ausübt.

$$\begin{aligned} \mathbf{F}(t) &= m\boldsymbol{\alpha}_1(t)\,, & (5.5)\\ \mathbf{D}(t) &= \mathbf{r}_1 \times \mathbf{F}(t) = \mathbf{I}_1\boldsymbol{\alpha}_1(t) + \boldsymbol{\omega}_1(t) \times \mathbf{I}_1\boldsymbol{\omega}_1(t)\,, & (5.6) \end{aligned}$$

wobei $\mathbf{a}_1$ und $\boldsymbol{\alpha}_1$ die lineare bzw. Winkelbeschleunigung von Körper 1 bezeichnen.

Da im impulsbasierten Kollisionsmodell Inertialkräfte vernachlässigt werden und m_1, $\mathbf{I}_1$ und $\mathbf{r}_1$ als konstant angenommen werden können, erhalten wir den linearen Impuls $\mathbf{P}$ sowie den Drehimpuls $\mathbf{L}$ den Körper 2 auf Körper 1 ausübt, indem wir $\mathbf{F}(t)$ und $\mathbf{D}(t)$ über die Kollisionsdauer t_f integrieren. Da eine Impulsübertragung ausschließlich im Kontaktpunkt stattfindet, entspricht der Gesamtimpuls $\mathbf{P}$ auf den Körper, dem Impuls $\mathbf{p}$ im Kontaktpunkt.

$$\begin{aligned} \mathbf{P}(t) &= \mathbf{p}(t) = \int_0^{t_f} \mathbf{F}(\tau)\,d\tau = m_1\Delta\boldsymbol{\nu}_1(t_f)\,, & (5.7)\\ \mathbf{l}(t) &= \int_0^{t_f} \mathbf{D}(\tau)\,d\tau = \mathbf{r}_1 \times \mathbf{p}(t_f) = \mathbf{I}_1\Delta\boldsymbol{\omega}(t_f)\,. & (5.8) \end{aligned}$$

Es erscheint daher zunächst naheliegend, die Kollisionszeit t als Integrationsparameter zu wählen und über die zeitliche Geschwindigkeitsänderung von Körper 1, den auf ihn ausgeübten Impuls zu ermitteln. Da wir in unserem Modell jedoch von einer infinitesimal kleinen Kollisionsdauer ausgehen, wird die zeitparametrisierte Geschwindigkeit zu einer unstetigen Funktion. Es ist somit offensichtlich, daß die Kollisionszeit ungeeignet ist, die Veränderung der für unsere Berechnung relevanten Größen während der Kollision zu parametrisieren. In Abschnitt 5.5 werden wir das Problem der Parameterwahl aufgreifen und verschiedene Größen hinsichtlich ihrer Eignung als Integrationsparameter diskutieren.

Zunächst wollen wir jedoch annehmen, wir hätten einen Parameter γ gefunden, der die folgenden Eigenschaften erfüllt:

(i) Strenge Monotonie in t.

(ii) Alle durch γ parametrisierten Größen sind stetige Funktionen im Intervall $[\gamma_s, \gamma_f]$.

Dann können wir unter Berücksichtigung von 5.5 und 5.6 die Geschwindigkeitsänderung von Körper 1 im Kontaktpunkt folgendermaßen darstellen:

$$\begin{aligned} \Delta\mathbf{u}_1(\gamma) &= \Delta\boldsymbol{\nu}_1(\gamma) + \Delta\boldsymbol{\omega}_1(\gamma) \times \mathbf{r}_1\\ &= \frac{1}{m_1}\mathbf{p}(\gamma) + \left[\mathbf{I}_1^{-1}\big(\mathbf{r}_1 \times \mathbf{p}(\gamma)\big)\right] \times \mathbf{r}_1\\ &= \left[\frac{1}{m_1}\mathbf{E} - \mathbf{r}_1^{\times}\mathbf{I}_1^{-1}\mathbf{r}_1^{\times}\right]\mathbf{p}(\gamma)\,, \end{aligned}$$

wobei $\mathbf{E}$ die Einheitsmatrix und $\mathbf{r}_i^\times$ die Kreuzproduktmatrix bezüglich $\mathbf{r}_i$ ist. Analog erhält man für Körper 2:

$$\Delta\mathbf{u}_2(\gamma) \;=\; -\left[\frac{1}{m_2}\mathbf{E} - \mathbf{r}_2^\times \mathbf{I}_2^{-1}\mathbf{r}_2^\times\right]\mathbf{p}(\gamma)\,.$$

Betrachtet man nun die relative Kontaktpunktgeschwindigkeit $\mathbf{u}(\gamma) := \mathbf{u}_1(\gamma) - \mathbf{u}_2(\gamma)$, dann ergibt sich eine lineare Beziehung zwischen dem auf Körper 1 ausgeübten Kollisionsimpuls $\mathbf{p}$ und der Änderung von $\mathbf{u}$.

$$\Delta\mathbf{u}(\gamma) = \mathbf{K}\mathbf{p}(\gamma)\,, \tag{5.9}$$

wobei $\mathbf{K} := \left[\left(\frac{1}{m_1}+\frac{1}{m_2}\right)\mathbf{E} - \left(\mathbf{r}_1^\times \mathbf{I}_1^{-1}\mathbf{r}_1^\times + \mathbf{r}_2^\times \mathbf{I}_2^{-1}\mathbf{r}_2^\times\right)\right]$ eine reelle 3×3 Matrix ist.

Satz 5.1 (Eigenschaften der Kollisionsmatrix). *Für eine gegebene Kollision erfüllt die Kollisionsmatrix* $\mathbf{K}$ *die folgenden Eigenschaften:*

(i) Konstanz,

(ii) Symmetrie,

(iii) Positiv-Definitheit,

(iv) Regularität.

Beweis. Untersuchen wir die nachzuweisenden Eigenschaften in obiger Reihenfolge.

(i) Die Matrix ergibt sich aus den Massen, Trägheitsmatrizen und der relativen Lage des jeweiligen Massenschwerpunkts zum Kontaktpunkt. Diese Größen sind, wie wir bereits oben erwähnt haben, in unserem Modell konstant.

(ii) Sei $\mathbf{A}_i := -\mathbf{r}_i^\times \mathbf{I}_i^{-1}\mathbf{r}_i^\times$ für $i = 1, 2$, dann gilt aufgrund der Symmetrie der Trägheitsmatrix $\mathbf{I}$ sowie der Schiefsymmetrie der Kreuzproduktmatrix $\mathbf{r}_i^\times$:

$$\mathbf{A}_i = \mathbf{A}_i^T$$

und weiter

$$\mathbf{K}^T = \left[\left(\frac{1}{m_1}+\frac{1}{m_2}\right)\mathbf{E} + \mathbf{A}_1 + \mathbf{A}_2\right]^T = \mathbf{K}\,.$$

Somit ist $\mathbf{K}$ symmetrisch.

(iii) Wir wollen zunächst zeigen, daß $\mathbf{x}^T\mathbf{A}_i\mathbf{x} \geq 0$ für alle $\mathbf{x} \neq 0$ gilt, d.h. daß $\mathbf{A}_i$ positivsemidefinit ist.
Berücksichtigen wir, daß $\mathbf{r}_i^\times$ eine schiefsymmetrische Matrix ist, so sehen wir für alle $\mathbf{x} \neq 0$:

$$\begin{aligned}
\mathbf{x}^T\mathbf{A}_i\mathbf{x} &= -\mathbf{x}^T\mathbf{r}_i^\times\mathbf{I}_i^{-1}\mathbf{r}_i^\times\mathbf{x} \\
&= \left(\mathbf{r}_i^\times\mathbf{x}\right)^T\mathbf{I}_i^{-1}\left(\mathbf{r}_i^\times\mathbf{x}\right) \\
&= \mathbf{y}^T\mathbf{I}_i^{-1}\mathbf{y} \qquad \text{mit } \mathbf{y} := \mathbf{r}_i^\times\mathbf{x} \\
&\geq 0\,,
\end{aligned}$$

da $\mathbf{I}_i$ und somit auch $\mathbf{I}_i^{-1}$ positiv-definit ist.
Die nachzuweisende Eigenschaft folgt nun unmittelbar aus:

$$\begin{aligned}\mathbf{x}^T\mathbf{K}\mathbf{x} &= \mathbf{x}^T\left[\left(\frac{1}{m_1}+\frac{1}{m_2}\right)\mathbf{E}+\mathbf{A}_1+\mathbf{A}_2\right]\mathbf{x}\\ &= \underbrace{\left(\frac{1}{m_1}+\frac{1}{m_2}\right)\mathbf{x}^T\mathbf{x}}_{>0}+\underbrace{\mathbf{x}^T\mathbf{A}_1\mathbf{x}}_{\geq 0}+\underbrace{\mathbf{x}^T\mathbf{A}_2\mathbf{x}}_{\geq 0}\\ &> 0 \qquad \text{für alle } \mathbf{x}\neq\mathbf{0}\,.\end{aligned}$$

(iv) Die Regularitätseigenschaft folgt unmittelbar aus (iii).

□

Im folgenden wird es sich gelegentlich als nützlich erweisen, die Zeilenvektoren von $\mathbf{K}$ mit $\mathbf{k}_x$, $\mathbf{k}_y$ und $\mathbf{k}_z$ zu bezeichnen.

Satz 5.2 (Kollisionsgleichungen). *Während einer durch γ parametrisierten Kollision stehen die relative Kontaktpunktgeschwindigkeit und der Kollisionsimpuls in folgender Beziehung:*

$$\frac{d}{d\gamma}\mathbf{u} = \mathbf{K}\frac{d}{d\gamma}\mathbf{p}\,, \tag{5.10}$$

$$\frac{d}{d\gamma}\mathbf{p} = \mathbf{K}^{-1}\frac{d}{d\gamma}\mathbf{u}\,. \tag{5.11}$$

Beweis. Berücksichtigen wir Resultat 5.9 sowie die Konstanz der Kollisionsmatrix $\mathbf{K}$, dann führt Differentiation bezüglich γ zur ersten Gleichung.
Die Invertierbarkeit von $\mathbf{K}$ liefert schließlich die zweite Beziehung. □

5.4 Kollisionsintegration

Die im vorangegangenen Abschnitt hergeleiteten Kollisionsgleichungen stellen ein wichtiges Ergebnis dar, da sie aufzeigen, wie der von uns gesuchte Kollisionsimpuls berechnet werden kann. Dazu ist es lediglich notwendig, eine Beschreibung der Änderung von $\mathbf{u}$ bezüglich des Kollisionsparameters γ zu finden, die es uns erlaubt, $\mathbf{u}$ während der gesamten Kollision zu verfolgen. Da der Impuls $\mathbf{p}$ in einer linearen Beziehung zur Änderung von $\mathbf{u}$ steht, können wir anhand von 5.11 zu jedem „Zeitpunkt"der Kollision den bisher übertragenen Impuls ermitteln. Wir wollen also im folgenden überlegen, wie sich die relative Kontaktpunktgeschwindigkeit während der Kollision verhält.

5.4.1 Integration von u in der Normalenrichtung

Betrachten wir zunächst die Komponente von $\mathbf{u}$ in Normalenrichtung. Für jede Kollision im physikalischen Sinne muß gelten, daß bei Eintritt der Kollision $u_z < 0$ ist, d.h. daß die beiden Körper sich aufeinanderzu bewegt haben. Da die Kontaktkräfte

eine Durchdringung verhindern, wird u_z am Ende der Kollision nichtnegativ sein. Die relative Kontaktpunktgeschwindigkeit in Normalenrichtung wird umso größer sein, je elastischer der Stoß war, d.h. je mehr gespeicherte Energie an die Körper zurückgegeben wurde.

Der Übergang zwischen der Kompressions- und der Rückstellungsphase ist für unsere Berechnung von großer Bedeutung, da wir erst im Zeitpunkt maximaler Kompression mittels der Strongeschen Hypothese 5.1 den Endzeitpunkt der Kollisionsintegration festlegen können.

5.4.2 Integration von u in der Tangentialebene

Wenden wir uns nun der Frage zu, wie sich $\mathbf{u}$ in der Tangentialebene verhalten wird. Die Änderung von $\mathbf{u_t}$ wird bestimmt durch die während der Kollision auftretenden Reibungskräfte. Wir müssen daher eine Fallunterscheidung treffen, die sich an den Einzelaussagen des Reibungsmodells orientiert.

Gleitbewegung

Die an der Kollision beteiligten Körper gleiten relativ zueinander, wenn die relative Kontaktpunktgeschwindigkeit $\mathbf{u}$ nicht verschwindet.

Definition 5.2 (Gleitrichtung und Gleitvektor). Sei $\mathbf{u_t} > 0$ die relative Kontaktpunktgeschwindigkeit in der Tangentialebene zu einem festen Parameterwert während der Kollision. Dann nennen wir den Winkel

$$\theta := \arctan(u_y, u_x)$$

relative Gleitrichtung von Körper 1 auf Körper 2. Als Gleitvektor bezüglich der Gleitrichtung θ fassen wir den Vektor $\xi(\theta)$ auf, der folgendermaßen definiert ist:

$$\xi(\theta) := \begin{bmatrix} -\mu\cos(\theta) \\ -\mu\sin(\theta) \\ 1 \end{bmatrix} = \begin{bmatrix} -\mu u_x/\sqrt{u_x^2+u_y^2} \\ -\mu u_y/\sqrt{u_x^2+u_y^2} \\ 1 \end{bmatrix} .$$

□

Lemma 5.3. *Sei $\theta(\gamma)$ die relative Gleitrichtung während der Kollision. Solange die Körper relativ zueinander gleiten, gilt:*

$$\frac{d}{d\gamma}\mathbf{p}(\gamma) = \xi\big(\theta(\gamma)\big)F_z(\gamma)\frac{dt}{d\gamma}(\gamma) . \tag{5.12}$$

Beweis. Aufgrund der Monotonieeigenschaft des Kollisionsparameters gilt:

$$\begin{aligned} \frac{d}{d\gamma}\mathbf{p}(\gamma) &= \frac{d}{dt}\mathbf{p}(\gamma)\frac{dt}{d\gamma}(\gamma) \\ &= \mathbf{F}(\gamma)\frac{dt}{d\gamma}(\gamma) . \end{aligned}$$

Dabei bezeichnet $\mathbf{F}(\gamma)$ die während der Kollision wirkende Gesamtkraft, d.h. die Zeitableitung des Kollisionsimpulses.
Nach dem Coulombschen Reibungsmodell gilt für den Gleitreibungsfall:

$$\begin{aligned}
\mathbf{F}(\gamma) &= -\mu\|\mathbf{F}_n(\gamma)\|\hat{\mathbf{u}}_t + \mathbf{F}_n(\gamma) \\
&= -\mu F_z(\gamma)\begin{bmatrix}\cos\big(\theta(\gamma)\big)\\ \sin\big(\theta(\gamma)\big)\\ 0\end{bmatrix} + \begin{bmatrix}0\\0\\F_z(\gamma)\end{bmatrix} \\
&= \begin{bmatrix}-\mu\cos\big(\theta(\gamma)\big)\\ -\mu\sin\big(\theta(\gamma)\big)\\ 1\end{bmatrix} F_z(\gamma) \\
&= \xi\big(\theta(\gamma)\big)F_z(\gamma)\,.
\end{aligned}$$

Damit folgt nun:

$$\frac{d}{d\gamma}\mathbf{p}(\gamma) = \xi\big(\theta(\gamma)\big)F_z(\gamma)\frac{dt}{d\gamma}(\gamma)\,.$$

□

Satz 5.4 (Ableitung von u im Gleitreibungsfall). *Gleiten zwei kollidierende Körper relativ zueinander, d.h. ist* $\mathbf{u_t} > 0$*, dann lautet die Ableitung der relativen Kontaktpunktgeschwindigkeit* $\mathbf{u}$ *bezüglich des Parameters* γ *folgendermaßen:*

$$\frac{d}{d\gamma}\mathbf{u} = \mathbf{K}\xi(\theta)F_z\frac{dt}{d\gamma} = K\begin{bmatrix}-\mu u_x/\sqrt{u_x^2+u_y^2}\\ -\mu u_y/\sqrt{u_x^2+u_y^2}\\ 1\end{bmatrix} F_z\frac{dt}{d\gamma}\,. \tag{5.13}$$

Beweis. Die Aussage folgt unmittelbar aus 5.10 und 5.12. □

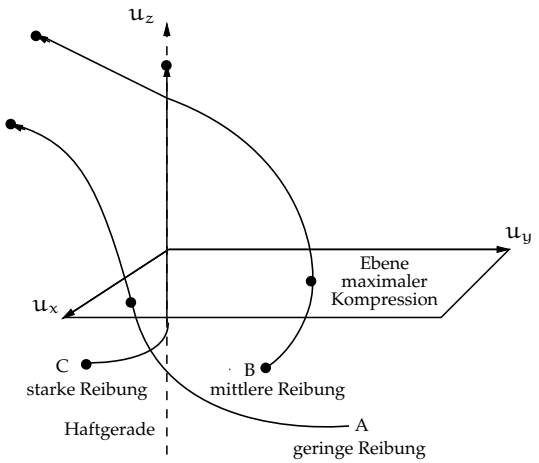
Durch Integration dieser nichtlinearen Differentialgleichung in Abhängigkeit vom Kollisionsparameter γ können wir $\mathbf{u}$ in der Gleitphase verfolgen. *Pfad A* in Abbildung 5.3 beschreibt beispielhaft die Veränderung von $\mathbf{u}$ während einer Kollision, die vollständig als Gleitbewegung erklärt werden kann, d.h. während der die Reibungskräfte nicht stark genug sind, um $\mathbf{u_t}$ verschwinden zu lassen.

Haftzustand

Die in 5.4 ermittelte Gleichung hat Gültigkeit, solange die Körper aneinander entlanggleiten. Diese Bedingung ist jedoch verletzt, wenn die relative Kontaktpunktgeschwindigkeit für einen Parameterwert γ verschwindet. In Abbildung 5.3 bedeutet dies, daß $\mathbf{u}(\gamma)$ auf der u_z-Achse, der sogenannten Haftgeraden liegt.

Bleibt die Haftbedingung bis zum Endzeitpunkt der Kollision gültig, dann bezeichnet man diesen Zustand als *stabiles Haften*. Dies bedeutet, daß $\mathbf{u}(\gamma)$ die Haftgerade in Abbildung 5.3 während der Kollision nicht mehr verlassen wird (*Pfad C*).

Abbildung 5.3 Der Verlauf der relativen Kontaktpunktgeschwindigkeit für Kollisionen mit unterschiedlich starken Reibungskräften.



Die auftretenden Reibungskräfte sind jedoch nicht immer stark genug, die Haftbedingung des Coulombschen Reibungsgesetzes 5.3 aufrechtzuerhalten (*instabile Haftung*). In [BK94, BK96a, BK96b] sowie in [Mir96b] wird anhand eines rein algebraischem bzw. eines eher geometrischem Beweises gezeigt, daß im Fall instabiler Haftung die Richtung β, in der die Gleitbewegung fortgesetzt wird, durch die Kollisionsmatrix eindeutig bestimmt und während der verbleibenden Kollisionsdauer konstant ist. Somit kann die Integration von $\mathbf{u}$ bezüglich γ mittels Gleichung 5.4 weitergeführt werden. *Pfad B* beschreibt den Fall instabiler Haftung, in dem $\mathbf{u}(\gamma)$ die Haftgerade in β-Richtung verläßt. Wir müssen also zwei Fragen beantworten

(i) Welche Bedingungen müssen erfüllt sein, damit ein Verschwinden der relativen Kontaktpunktgeschwindigkeit mit stabiler Haftung einhergeht?

(ii) Wie wird die Gleitbewegung im Fall instabiler Haftung fortgesetzt werden?

Wir wollen uns zunächst Frage (i) zuwenden.

Lemma 5.5 (Monotonie von u_z im Fall stabiler Haftung). *Sei* $\mathbf{K}$ *die Kollisionsmatrix der betrachteten Kollision, während der im Parameterwert* $\overline{\gamma}$ *stabile Haftung auftritt. Dann wächst die Relativgeschwindigkeit im Kontaktpunkt bis zu Ende der Kollision streng monoton, d.h.*

$$\frac{du_z}{d\gamma}(\gamma) > 0 \qquad \textit{für } \gamma \in [\overline{\gamma}, \gamma_f) \, .$$

Beweis. Im Fall stabiler Haftung verschwindet $(d\mathbf{u_t}/d\gamma)(\gamma)$ für $\gamma \in [\overline{\gamma}, \gamma_f)$, d.h. die Relativgeschwindigkeit in der Tangentialebene bleibt im weiteren Kollisionsverlauf un-

verändert. Somit kann die Kollisionsgleichung 5.11 weiter vereinfacht werden.

$$\begin{aligned}\frac{d}{d\gamma}\mathbf{p}(\gamma) &= \mathbf{K}^{-1}\frac{d}{d\gamma}\mathbf{u}(\gamma)\\ &= \mathbf{K}^{-1}\begin{bmatrix}0\\0\\\frac{du_z}{d\gamma}(\gamma)\end{bmatrix}.\end{aligned}$$

Somit gilt:

$$\frac{dp_z}{d\gamma}(\gamma) = K_{33}^{-1}\frac{du_z}{d\gamma}(\gamma)\,.$$

Die inverse Kollisionsmatrix kann folgendermaßen dargestellt werden:

$$\mathbf{K}^{-1} = \frac{ad(\mathbf{K})}{\det(\mathbf{K})}\,,$$

wobei $ad(\mathbf{K})$ die adjungierte Matrix zu $\mathbf{K}$ ist.
Somit gilt für K_{33}^{-1}:

$$K_{33}^{-1} = \frac{\det\left(\mathbf{K}^{(3,3)}\right)}{\det(\mathbf{K})}\,,$$

wobei $\mathbf{K}^{(3,3)}$ die Matrix ist, die aus $\mathbf{K}$ durch Löschen der dritten Zeile und Spalte hervorgeht.
$\mathbf{K}^{(3,3)}$ ist als Hauptuntermatrix von $\mathbf{K}$ ebenfalls positiv-definit, so daß $\det\left(\mathbf{K}^{(3,3)}\right) > 0$ und weiter $K_{33}^{-1} > 0$ gilt.
Aus obiger Beziehung folgt daher:

$$\begin{aligned}\frac{du_z}{d\gamma}(\gamma) &= \frac{1}{K_{33}^{-1}}\frac{dp_z}{d\gamma}(\gamma)\\ &= \frac{1}{K_{33}^{-1}}\frac{dp_z}{dt}(\gamma)\frac{dt}{d\gamma}(\gamma)\\ &= \frac{1}{K_{33}^{-1}}F_z(\gamma)\frac{dt}{d\gamma}(\gamma) \qquad (5.14)\\ &> 0\,,\end{aligned}$$

da die Kontaktkräfte im Intervall $[\gamma_s, \gamma_f)$ positiv sind und der Kollisionsparameter γ die Monotonieeigenschaft erfüllt. □

Satz 5.6 (Stabile Haftung). *Sei* $\mathbf{K}$ *die Kollisionsmatrix der betrachteten Kollision. Die Reibungskräfte können die Haftung bis zum Ende der Kollision genau dann aufrechterhalten, wenn gilt:*

$$\left(K_{13}^{-1}\right)^2 + \left(K_{23}^{-1}\right)^2 \leq \mu^2\left(K_{33}^{-1}\right)^2, \qquad (5.15)$$

wobei K_{ij}^{-1} *den Eintrag* (i,j) *der inversen Kollisionsmatrix* $\mathbf{K}^{-1}$ *bezeichnet.*

Beweis. Sei $\overline{\gamma}$ der Parameterwert, für den gilt, daß $\mathbf{u}_t(\overline{\gamma})$ verschwindet und sei $\mathbf{F}(\overline{\gamma})$ die Gesamtkraft, die auf Körper 1 im Parameterwert $\overline{\gamma}$ wirkt. Nach dem Coulombschen Reibungsgesetz 5.3 gilt nun:

$$\begin{aligned} & \|\mathbf{F}_t(\overline{\gamma})\| \leq \mu \|\mathbf{F}_n(\overline{\gamma})\| \\ \Longleftrightarrow \quad & F_x(\overline{\gamma})^2 + F_y(\overline{\gamma})^2 \leq \mu^2 F_z(\overline{\gamma})^2 \\ \Longleftrightarrow \quad & \left[\frac{dp_x}{dt}(\overline{\gamma})\right]^2 + \left[\frac{dp_y}{dt}(\overline{\gamma})\right]^2 \leq \mu^2 \left[\frac{dp_z}{dt}(\overline{\gamma})\right]^2. \end{aligned}$$

Multipliziert man obige Gleichung mit $(dt/d\gamma)(\overline{\gamma})$, dann erhält man:

$$\left[\frac{dp_x}{d\gamma}(\overline{\gamma})\right]^2 + \left[\frac{dp_y}{d\gamma}(\overline{\gamma})\right]^2 \leq \mu^2 \left[\frac{dp_z}{d\gamma}(\overline{\gamma})\right]^2.$$

Unter Verwendung von 5.11 läßt sich die Ungleichung folgendermaßen darstellen:

$$\left[k_x^{-1}\frac{d\mathbf{u}}{d\gamma}(\overline{\gamma})\right]^2 + \left[k_y^{-1}\frac{d\mathbf{u}}{d\gamma}(\overline{\gamma})\right]^2 \leq \mu^2 \left[k_z^{-1}\frac{d\mathbf{u}}{d\gamma}(\overline{\gamma})\right]^2.$$

Ist die Haftung stabil, dann verschwindet $(d\mathbf{u}_t/d\gamma)$, so daß gilt:

$$\left[K_{13}^{-1}\frac{du_z}{d\gamma}(\overline{\gamma})\right]^2 + \left[K_{23}^{-1}\frac{du_z}{d\gamma}(\overline{\gamma})\right]^2 \leq \mu^2 \left[K_{33}^{-1}\frac{du_z}{d\gamma}(\overline{\gamma})\right]^2.$$

Lemma 5.5 erlaubt schließlich die Division der Ungleichung durch $(du_z/d\gamma)(\overline{\gamma})$, womit unsere Behauptung bewiesen ist. □

Die Problematik von Frage (ii) ergibt sich unmittelbar aus der Aussage des Coulombschen Reibungsgesetzes für den Haftreibungsfall 5.3. Das Gesetz erlaubt lediglich die Reibungskraft nach oben abzuschätzen, so daß diese durch unser Modell nicht vollständig bestimmt ist. Die Antwort auf Frage (ii) erfordert daher eine weitergehende Analyse und Interpretation der Kollisionsgleichungen im Gleit- und Haftreibungsfall, weswegen wir dem Leser das Studium der Arbeiten von MIRTICH [Mir96b] und BHATT und KOECHLING [BK94, BK96a, BK96b] empfehlen. Die zentralen Ergebnisse wollen wir jedoch aus Gründen der Vollständigkeit vorstellen.

Definition 5.3 (Strahlen konstanter Gleitrichtung). Zeigt der Vektor der relativen Kontaktpunktgeschwindigkeit $\mathbf{u}$ sowie dessen Ableitung $\frac{d\mathbf{u}}{d\gamma}$ in die gleiche bzw. entgegengesetzte Richtung, dann nennen wir den durch $\theta = \arctan(u_y, u_x)$ definierten Ursprungsstrahl einen Strahl konstanter Gleitrichtung.

Gilt im speziellen $\arctan\left(\frac{du_y}{d\gamma}, \frac{du_x}{d\gamma}\right) = \theta$, so sprechen wir von einem sogenannten *Divergenzstrahl*. □

In der Arbeit von MIRTICH werden die folgenden Aussagen bewiesen, die wir an dieser Stelle zu einem Satz zusammengefaßt haben.

Satz 5.7 (Eindeutigkeit des Divergenzstrahls). *Im Falle stabiler Haftung gibt es keine Divergenzstrahlen. Ist die Haftung dagegen instabil, dann existiert genau ein Divergenzstrahl, der durch den Winkel β mit folgenden Eigenschaften eindeutig bestimmt ist.*

(i) $\big(\hat{\mathbf{u}} \times (d\mathbf{u}/d\gamma)\big)(\beta) = 0 \quad \Longleftrightarrow$
$(K_{11} - K_{12})\mu \cos\beta \sin\beta + K_{12}\mu(\sin^2\beta - \cos^2\beta) + K_{32}\cos\beta - K_{13}\sin\beta = 0$

(ii) $\big(\hat{\mathbf{u}}^T(d\mathbf{u}/d\gamma)\big)(\beta) \quad > 0 \quad \Longleftrightarrow$
$-K_{11}\mu\cos\beta - K_{22}\mu\sin^2\beta - 2K_{12}\mu\cos\beta\sin\beta + K_{32}\sin\beta + K_{13}\cos\beta > 0$

Durch Anwendung der Substitution $t := \tan\left(\frac{\beta}{2}\right)$ kann die Sinus- und Cosinusfunktion aus (i) und (ii) eliminiert werden. Die beiden Kriterien aus Satz 5.7 lauten nun:

(i') $(K_{12}\mu + K_{32})t^4 + 2\big[(K_{11} - K_{22})\mu + K_{13}\big]t^3 - 6K_{12}\mu t^2 + 2\big[(K_{22}\mu - K_{11})\mu + K_{13}\big]t +$
$(K_{12}\mu - K_{32}) = 0$

(ii') $-(K_{13} + K_{11}\mu)t^4 + (2K_{32} + 4K_{12}\mu)t^3 + \mu(2K_{11} - 4K_{22})t^2 + (2K_{32} - 4K_{12}\mu)t +$
$(K_{13} - K_{11}\mu) > 0$

Den gesuchten Winkel erhält man durch Rücksubstitution des Wertes t^*, der Bedingung (i') und (ii') erfüllt.

5.4.3 Integration von W_z – Anwendung der Strongeschen Hypothese

Eine offengebliebene Frage im Kontext der Kollisionsintegration ist die Bestimmung des Endzeitpunkts der Kollision. Wie wir bereits in 5.1.2 erwähnt haben, wollen wir hierzu eine einfache Regel, die sogenannte Strongesche Hypothese heranziehen, welche die von der Kollisionsgesamtkraft an den beiden Körpern verrichtete Arbeit in Normalenrichtung im Endzeitpunkt der Kollision sowie im Zeitpunkt maximaler Kompression zueinander in Beziehung setzt.

Im Zeitintervall $[0, t_f]$ verrichtet die Kollisionskraft an Körper 1 im Kontaktpunkt die folgende Arbeit (dabei nehmen wir an, daß nahester Punkt q_1 und Kontaktpunkt zusammenfallen):

$$W_{z1}(t_f) = \int_0^{t_f} F_z(t)\dot{\mathbf{q}}_{z1}(t)\,dt\,.$$

Parametrisiert man die Kollision mit einem geeigneten Integrationsparameter γ, so erhält man die von dem Kollisionsimpuls an Körper 1 verrichtete Arbeit als

$$\begin{aligned} W_{z1}(\gamma_f) &= \int_{\gamma_s}^{\gamma_f} F_z(\gamma)u_{z1}(\gamma)\frac{dt}{d\gamma}\,d\gamma \\ &= \int_{\gamma_s}^{\gamma_f} \frac{dp_z}{d\gamma}u_{z1}(\gamma)\,d\gamma\,. \end{aligned}$$

Analog gilt für Körper 2:

$$W_{z2}(\gamma_f) = -\int_{\gamma_s}^{\gamma_f} \frac{dp_z}{d\gamma}u_{z2}(\gamma)\,d\gamma\,.$$

Somit erhalten wir die Gesamtarbeit in Normalenrichtung, welche die Kollisionsimpulse an den beiden Körpern verrichten als

$$W_z(\gamma_f) = \int_{\gamma_s}^{\gamma_f} \frac{dp_z}{d\gamma} u_z(\gamma)\, d\gamma\,, \tag{5.16}$$

sowie ihre Ableitung zu einem beliebigen Parameterwert γ:

$$\begin{aligned} \frac{dW_z}{d\gamma}(\gamma) &= \frac{dp_z}{d\gamma} u_z(\gamma) \\ &= u_z(\gamma) \mathbf{k}_z^{-1} \frac{d}{d\gamma}\mathbf{u}(\gamma)\,. \end{aligned} \tag{5.17}$$

Diese Gleichung erlaubt uns, die von den Kollisionsimpulsen verrichtete Arbeit während der Kollision mittels numerischer Integration zu berechnen.

Da wir neben W_z auch u_z verfolgen, können wir auf einfache Weise den Zeitpunkt maximaler Kompression erkennen. Es gilt die folgende Bedingung:

$$u_z(\gamma_{mc}) = 0\,.$$

Nun können wir die Integration unterbrechen und die Strongesche Hypothese anwenden, die uns eine Abbruchbedingung für die Integration der Rückstellungsphase und somit der gesamten Impulsberechnung liefert:

$$W_z(\gamma_f) = (1 - e^2) W_z(\gamma_{mc})$$

Nachdem wir $W_z(\gamma_f)$ ermittelt haben, setzen wir die Kollisionsintegration fort, und zwar bis W_z den Endwert erreicht hat.

5.4.4 Zusammenfassung

Kollisionsintegration bedeutet das Verfolgen der relativen Kontaktpunktgeschwindigkeit $\mathbf{u}(\gamma)$. Reibungskräfte beeinflussen die Geschwindigkeitsänderung in der Tangentialebene, so daß wir entsprechend dem Coulombschen Modell Gleiten und Haften als Kontaktform unterscheiden müssen. Das Coulombsche Reibungsgesetz führt im Gleitreibungsfall zu einer Differentialgleichung 1. Ordnung, die wir numerisch integrieren müssen. Haften die Körper stabil aufeinander, dann vereinfacht sich die Gleichung für die Änderung von $\mathbf{u}$. Nach Satz 5.4 und unter Berücksichtigung von 5.14 gilt:

(i)

$$\frac{d}{d\gamma}\mathbf{u} = \begin{cases} \mathbf{K}\xi(\theta) F_z \frac{dt}{d\gamma} & \text{falls } \mathbf{u_t} \neq 0\,; \\ \frac{1}{K_{33}^{-1}} \mathbf{e}_z F_z \frac{dt}{d\gamma} & \text{falls } \mathbf{u_t} = 0 \text{ und } 5.15\,; \\ \mathbf{K}\xi(\beta) F_z \frac{dt}{d\gamma} & \text{sonst}\,. \end{cases} \tag{5.18}$$

(ii)

$$\frac{dW_z}{d\gamma} = u_z \frac{dp_z}{d\gamma}\,. \tag{5.19}$$

Dabei ist $\mathbf{e}_z$ der Einheitsvektor in Normalenrichtung und β die eindeutig bestimmte, konstante Richtung, in der die Gleitbewegung im Fall instabiler Haftung fortgesetzt wird.

Auf diese Weise können wir $\mathbf{u}$ bis zum Zeitpunkt maximaler Kompression verfolgen. Nun erlaubt uns die Hypothese von STRONGE den Endpunkt γ_f der Kollision zu bestimmen. Die sich anschließende Integration der Rückstellungsphase liefert $\mathbf{u}(\gamma_f)$, d.h. die relative Kontaktpunktgeschwindigkeit am Ende der Kollision. Der Gesamtimpuls $\mathbf{p}(\gamma_f)$, der während der Kollision von Körper 2 auf Körper 1 im Kontaktpunkt übertragen wird, kann nun anhand von 5.11 ermittelt werden:

$$\mathbf{p}(\gamma_f) = \mathbf{K}^{-1}\Delta\mathbf{u}(\gamma_f) = \mathbf{K}^{-1}\left[\mathbf{u}(\gamma_f) - \mathbf{u}(\gamma_s)\right] .$$

5.5 Kollisionsparametrisierung

Die Integration der Kollisionsgleichung 5.11 erfordert die Wahl eines geeigneten Kollisionsparameters γ. In Abschnitt 5.4 haben wir argumentiert, daß die natürlichste Größe in diesem Zusammenhang, nämlich die Kollisionsdauer, eine denkbar ungünstige Wahl darstellt, weil das Parameterintervall, über welches die Kollisionsgleichung zu integrieren wäre, infinitesimal klein ist. Daher wollen wir zunächst überlegen, welche Größen entsprechend den von uns in 5.4 definierten Anforderungen als Kollisionsparameter geeignet wären.

5.5.1 p_z-Parametrisierung

Eine günstige Wahl des Kollisionsparameters γ ist der Kollisionsimpuls in Normalenrichtung, welcher im Kollisionskoordinatensystem mit der Komponente p_z von $\mathbf{p}$ zusammenfällt.

In dem von uns betrachteten Kollisionsmodell ist die einzige Kraft, die in diese Richtung wirkt, die Kontaktkraft, welche eine Durchdringung der an der Kollision beteiligten Körper verhindert. Legt man daher das Kollisionskoordinatensystem der Betrachtung zugrunde, dann ist die Komponente F_z der Gesamtkraft im Intervall $[t, t_f)$ stets positiv und p_z als Integral über diese Kraftkomponente streng monoton wachsend über die Kollisionsdauer. Da p_z die geforderte Monotonieeigenschaft erfüllt, müssen wir noch zeigen, daß $\mathbf{u}$ und W_z als zu integrierende Größen stetige Funktionen in p_z sind.

Satz 5.8 (Integration unter p_z-Parametrisierung). *Während einer durch p_z parametrisierten Kollision gilt für die Änderung der Relativgeschwindigkeit im Kontaktpunkt:*

(i)

$$\frac{d}{dp_z}\mathbf{u} = \begin{cases} \mathbf{K}\xi(\theta) & \text{falls } \mathbf{u_t} \neq \mathbf{0}\,; \\ \frac{1}{K_{33}^{-1}}\mathbf{e}_z = const & \text{falls } \mathbf{u_t} = \mathbf{0} \text{ und 5.15}\,; \\ \mathbf{K}\xi(\beta) = const & \text{sonst}\,. \end{cases} \tag{5.20}$$

(ii)

$$\frac{dW_z}{dp_z} = u_z\,, \tag{5.21}$$

Dabei ist $\mathbf{e}_z$ der Einheitsvektor in Normalenrichtung und β die eindeutig bestimmte, konstante Richtung, in der die Gleitbewegung im Fall instabiler Haftung fortgesetzt wird.

Beweis. Die Aussage im Gleitreibungsfall folgt unmittelbar aus Satz 5.4, wenn man γ durch p_z ersetzt.

$$\begin{aligned} \frac{d}{dp_z}\mathbf{u}(p_z) &= \mathbf{K}\xi\big(\theta(p_z)\big)F_z(p_z)\frac{dt}{dp_z}(p_z) \\ &= \mathbf{K}\xi\big(\theta(p_z)\big)F_z(p_z)\frac{1}{F_z(p_z)} \\ &= \mathbf{K}\xi\big(\theta(p_z)\big)\,. \end{aligned}$$

Im Fall stabiler Haftung unter p_z-Parametrisierung kann auch Gleichung 5.14 für $(du_z)/(d\gamma)$ weiter vereinfacht werden:

$$\frac{du_z}{dp_z} = \frac{1}{K_{33}^{-1}}F_z(p_z)\frac{dt}{dp_z}(p_z) = \frac{1}{K_{33}^{-1}}\,.$$

Ist die Haftung instabil, dann müssen wir die neue Gleitrichtung β sowie den korrespondierenden Gleitrichtungsvektor $\xi(\beta)$ berechnen. □

Korollar 5.9. Während der gesamten Kollision ist die relative Kontaktpunktgeschwindigkeit $\mathbf{u}$ eine stetige Funktion der Normalenkomponente des Kollisionsimpulses p_z.

Beobachtung 10 *(Lösung der Differentialgleichung im Haftungsfall).* Im Haftungsfall können die Differentialgleichungen aus Satz 5.8 geschlossen gelöst werden. Sei a der Wert von p_z zum Zeitpunkt des Haftungseintritts und b der Wert, den p_z am Ende der Integrationsphase annimmt, dann ergibt sich als Endwert der Integration für $\mathbf{u}$ und W_z:

$$\begin{aligned} \mathbf{u}(b) &= \begin{cases} \mathbf{u}(a) + \frac{b-a}{K_{33}^{-1}}\mathbf{e}_z & \text{falls 5.15}\,; \\ \mathbf{u}(a) + \mathbf{K}\xi(\beta)(b-a) & \text{sonst}\,. \end{cases} \\ W_z(b) = \int_a^b u_z(p_z)\,dp_z &= \begin{cases} (b-a)u_z(a) + \frac{(b-a)^2}{K_{33}^{-1}}\mathbf{e}_z & \text{falls 5.15}\,; \\ (b-a)u_z(a) + (b-a)^2\mathbf{k}_z\xi(\beta) & \text{sonst}\,. \end{cases} \end{aligned}$$

Offensichtlich haben wir eine Parametrisierung gefunden, die uns erlaubt, die Relativgeschwindigkeit und somit den übertragenen Impuls über die Kompressions- und Rückstellungsphase hinweg zu verfolgen. Das folgende, primär implementierungstechnische Problem haben wir bisher jedoch vernachlässigt. Wie wir bereits herausgestellt haben, ist es notwendig, die Kollisionsintegration zum Zeitpunkt der maximalen Kompression, d.h. wenn $u_z = 0$ gilt, zu unterbrechen, um mittels der Strongeschen Hypothese den Endzeitpunkt der Integration festzulegen. Parametrisieren wir jedoch die numerische Integration mittels p_z, dann steht zum Zeitpunkt des Aufrufs unseres Differentialgleichungslösers die obere Grenze des Integrationsintervalls weder für die Kompressions-, noch für die Rückstellungsphase fest. Daher sollten wir überlegen, ob nicht alternative Parameter existieren, die eine Charakterisierung beider Integrationsintervalle erlauben.

5.5.2 u_z-/W_z-Parametrisierung

Die Wahl von u_z und W_z als Parameter der Kompressions- bzw. Rückstellungsintegration erscheint vor dem Hintergrund obiger Überlegungen vielversprechend, da die zugehörigen Integrationsintervalle über diese Größe beschrieben werden können.

Kompressionsintegration Parameter u_z: $\left[u_z(t=0), 0\right]$

Rückstellungsintegration Parameter W_z: $\left[W_z(u_z=0), (1-e^2)W_z(u_z=0)\right]$

Zunächst ist jedoch zu zeigen, daß u_z und W_z gültige Parameter für die jeweilige Integrationsphase darstellen.

Lemma 5.10 (Gültigkeit der γ-Parametrisierung). *Parametrisiert man die Kollisionsintegration mittels einer Größe γ, so stellt dies eine gültige Parametrisierung dar, falls*

$$0 < \frac{d\gamma}{dp_z}(\gamma) < \infty \qquad \textit{für } \gamma \in [\gamma_s, \gamma_f)$$

gilt.

Beweis. Sei γ ein beliebiger Parameter. Wollen wir zeigen, daß γ entsprechend den in Abschnitt 5.4 definierten Anforderungen gültig ist, so können wir ausnutzen, daß diese Eigenschaft bereits für p_z bekannt ist.

(i) Monotonieeigenschaft:

$$\frac{d\gamma}{dt}(\gamma) = \frac{d\gamma}{dp_z}(\gamma)\frac{dp_z}{dt}(\gamma) = \underbrace{\frac{d\gamma}{dp_z}(\gamma)}_{>0}\underbrace{F_z}_{>0} > 0\,,$$

(ii) Stetigkeitseigenschaft der zu integrierenden Funktionen:

$$\begin{aligned}
-\infty < \frac{d}{d\gamma}u_x(\gamma) &= \frac{d}{dp_z}u_x(p_z)\frac{dp_z}{d\gamma}(\gamma) < \infty\,,\\
-\infty < \frac{d}{d\gamma}u_y(\gamma) &= \frac{d}{dp_z}u_y(p_z)\frac{dp_z}{d\gamma}(\gamma) < \infty\,,\\
-\infty < \frac{d}{d\gamma}u_z(\gamma) &= \frac{d}{dp_z}u_z(p_z)\frac{dp_z}{d\gamma}(\gamma) < \infty\,,\\
-\infty < \frac{dW_z}{d\gamma}(\gamma) &= u_z(p_z)\frac{dp_z}{d\gamma}(\gamma) \qquad < \infty\,.
\end{aligned}$$

Aus der Differenzierbarkeit von $\mathbf{u}$ und W_z folgt unmittelbar ihre Stetigkeit.

□

Satz 5.11 (Gültigkeit der u_z-/W_z-Parametrisierung). *Parametrisiert man die Kompressionsintegration über die Normalenkomponente der relativen Kontaktpunktgeschwindigkeit u_z und setzt die Integration in der Rückstellungsphase mit der verrichteten Arbeit in Normalenrichtung W_z als Parameter fort, so ist diese Parametrisierung gültig, falls*

(i) zu Beginn der u_z-parametrisierten Integration gilt:

$$\frac{du_z}{dp_z}(p_z) > 0 \qquad \textbf{und}$$

(ii) zu Beginn der W_z parametrisierten Integration

$$u_z > 0$$

ist.

Beweis. Da (du_z/dp_z) in der Kompressionsphase streng monoton wächst (vgl. [Mir96b]) folgt aus Bedingung (i) und Satz 5.8 die Behauptung für u_z.
W_z ist aufgrund von (ii) gültig für die Rückstellungsphase, da

$$0 < \frac{dW_z}{dp_z} = u_z(p_z) < \infty\,.$$

□

Die Kollisionsintegration mit u_z und W_z als Parameter setzt somit voraus, daß wir Bedingung (i) und (ii) sicherstellen können. Dies ist leider nicht generell möglich.

Auch wenn es paradox klingen mag, es lassen sich Situationen herbeiführen, in denen die beiden Kontaktpartner zu Beginn der Kollisionspartner ineinander hineinbeschleunigt werden (vgl. [Bar89]). Obwohl solche Konstellationen lediglich in bewußt konstruierten Simulation beobachtet worden sind, präsentiert MIRTICH in [Mir96b] folgende Lösung für das Problem. Da (du_z/dp_z) in der Kompressionsphase streng monoton wächst, setzt man zunächst p_z als Integrationsparameter ein, bis $(du_z/dp_z)(p_z) > 0$

ist. Diese Bedingung muß eintreten, da zu Beginn der Kompressionsintegration $u_z < 0$ und zum Zeitpunkt maximaler Kompression $u_z = 0$ gilt.

Auch Bedingung (ii) können wir nicht garantieren, solange wir die Integration wie bisher beschrieben durchführen. Beim Übergang zur W_z-Parametrisierung gilt nämlich im Zeitpunkt maximaler Kompression:

$$\frac{dW_z}{d\gamma}(\gamma_{mc}) = u_z(\gamma_{mc})\frac{dp_z}{d\gamma}(\gamma_{mc}) = 0\,,$$

da die Bedingung $u_z(\gamma_{mc}) = 0$ gerade diesen Zeitpunkt charakterisiert.
Dieses Problem können wir jedoch leicht umgehen, indem wir die u_z-parametrisierte Kollisionsgleichung über den Zeitpunkt maximaler Kompression hinweg integrieren. Dabei müssen wir jedoch darauf achten, daß der durch die Strongesche Hypothese definierte Endwert für W_z nicht überschritten wird, da dieser den Endzeitpunkt der Kollision festlegt.

Lemma 5.12. *Sei W_z^R die Normalenkomponente der Arbeit, die während der Rückstellungsphase an den beiden Körpern verrichtet wird. Dehnt man die Integration bezüglich u_z in diese Phase hinein aus, d.h. bis zu einem bestimmten Wert $\overline{u}_z$, wobei*

$$\overline{u}_z = \sqrt{2W_z^R\left(K_{33} - \mu\sqrt{K_{31}^2 + K_{32}^2}\right)} > 0 \tag{5.22}$$

gilt, so wird die bis zu diesem Parameterwert verrichtete Arbeit in Normalenrichtung den Wert W_z^R nicht überschreiten.

Beweis. Nehmen wir zunächst einmal an, es existiere eine Konstante c, so daß

$$\frac{dW_z}{du_z} \leq cu_z$$

gilt. Dann ist c positiv, da

$$\frac{dW_z}{du_z}(u_z) = u_z \underbrace{\frac{dp_z}{du_z}(u_z)}_{>0} > 0 \qquad \text{für } u_z \in (0, \overline{u}_z]\,.$$

Die Integration von W_z über dem Parameterintervall $[0, \overline{u}_z]$ für u_z liefert:

$$\Delta W_z \leq \frac{1}{2}c\overline{u}_z^2\,.$$

Da wir sicherstellen wollen, daß ΔW_z den Wert W_z^R nicht überschreitet, muß gelten:

$$\overline{u}_z \leq \sqrt{\frac{2W_z^R}{c}}$$

Um den Beweis zu vervollständigen, müssen wir im folgenden die Konstante c festlegen. Wir können uns dabei auf den Gleitreibungsfall beschränken, da im Haftreibungsfall keine Integration erforderlich ist.
Aufgrund von 5.19 und 5.20 gilt im Fall einer u_z-Parametrisierung:

$$\begin{aligned}\frac{dW_z}{du_z} &= \frac{u_z}{\mathbf{k}_z\xi(\theta)} = \frac{u_z}{-\mu\left[\frac{K_{31}u_x+K_{32}u_y}{\sqrt{u_x^2+u_y^2}}\right]+K_{33}} \\ &\leq \frac{u_z}{-\mu\frac{\sqrt{K_{31}^2+K_{32}^2}\sqrt{u_x^2+u_y^2}}{\sqrt{u_x^2+u_y^2}}+K_{33}} \\ &= \underbrace{\frac{1}{K_{33}-\mu\sqrt{K_{31}^2+K_{32}^2}}}_{c}u_z\,,\end{aligned}$$

denn

$$\begin{aligned}K_{31}u_x+K_{32}u_y &\leq \sqrt{(K_{31}u_x+K_{32}u_y)^2} \\ &= \sqrt{K_{31}^2u_x^2+K_{32}^2u_y^2+K_{32}^2u_x^2+K_{31}^2u_x^2\underbrace{-(K_{31}u_y-K_{32}u_x)^2}_{<0}} \\ &\leq \sqrt{K_{31}^2u_x^2+K_{32}^2u_y^2+K_{31}^2u_y^2+K_{32}^2u_x^2} \\ &= \sqrt{(K_{31}^2+K_{32}^2)(u_x^2+u_y^2)} \\ &= \sqrt{K_{31}^2+K_{32}^2}\sqrt{u_x^2+u_y^2}\,.\end{aligned}$$

□

Abbildung 5.4 beschreibt, wie der modifizierte Ansatz zur Kollisionsintegration die relevanten Größen in den beiden Phasen der Energieübertragung verfolgt.

Satz 5.13 (Kompressions- und Überbrückungsintegration unter u_z-Parametrisierung). *Während der durch u_z-parametrisierten Kompressions- und Rückstellungsphase einer Kollision gilt für die Ableitung der Relativgeschwindigkeit im Kontaktpunkt und der an den Körpern verrichteten Arbeit in Normalenrichtung:*

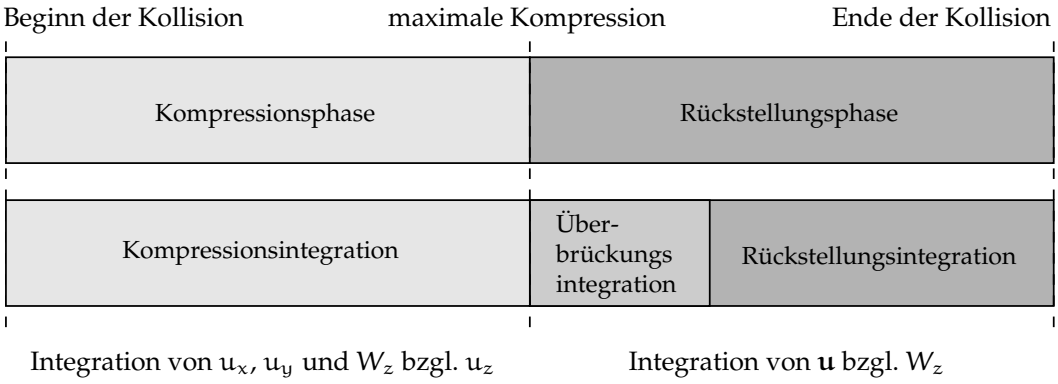
(i)

$$\frac{d}{du_z}\mathbf{u} = \begin{cases}\frac{1}{\mathbf{k}_z\xi(\theta)}K\xi(\theta) & \textit{falls } \mathbf{u_t} \neq 0\,; \\ \mathbf{e}_z = \textit{const} & \textit{falls } \mathbf{u_t} = 0 \textit{ und 5.15}\,; \\ \frac{1}{\mathbf{k}_z\xi(\beta)}K\xi(\beta) = \textit{const} & \textit{sonst}\,.\end{cases} \tag{5.23}$$

(ii)

$$\frac{dW_z}{du_z} = \begin{cases}\frac{1}{\mathbf{k}_z\xi(\theta)}u_z & \textit{falls } \mathbf{u_t} \neq 0\,; \\ K_{33}^{-1}u_z & \textit{falls } \mathbf{u_t} = 0 \textit{ und 5.15}\,; \\ \frac{1}{\mathbf{k}_z\xi(\beta)}u_z & \textit{sonst}\,.\end{cases} \tag{5.24}$$

Abbildung 5.4 Die erforderlichen Integrationen während den Kollisionsphasen.



Dabei ist $\mathbf{e}_z$ der Einheitsvektor in Normalenrichtung und β die eindeutig bestimmte, konstante Richtung, in der die Gleitbewegung im Fall instabiler Haftung fortgesetzt wird.

Beweis. Die Aussage bezüglich $\mathbf{u}$ folgt aus 5.18 unter Berücksichtigung von Satz 5.8.

$$\frac{\mathrm{d}}{\mathrm{du}_z}\mathbf{u} \overset{5.18}{=} \begin{cases} \mathbf{K}\boldsymbol{\xi}(\theta)\frac{\mathrm{dp}_z}{\mathrm{du}_z} \overset{5.20}{=} \mathbf{K}\boldsymbol{\xi}(\theta)\frac{1}{\mathbf{k}_z\boldsymbol{\xi}(\theta)} & \text{falls } \mathbf{u_t} \neq \mathbf{0}\,; \\ \frac{1}{K_{33}^{-1}}\mathbf{e}_z\frac{\mathrm{dp}_z}{\mathrm{du}_z} \overset{5.20}{=} \frac{1}{K_{33}^{-1}}\mathbf{e}_z K_{33}^{-1} = \mathbf{e}_z & \text{falls } \mathbf{u_t} = \mathbf{0} \text{ und } 5.15\,; \\ \mathbf{K}\boldsymbol{\xi}(\beta)\frac{\mathrm{dp}_z}{\mathrm{du}_z} \overset{5.20}{=} \mathbf{K}\boldsymbol{\xi}(\beta)\frac{1}{\mathbf{k}_z\boldsymbol{\xi}(\beta)} & \text{sonst}\,. \end{cases}$$

Für die an den beteiligten Körpern verrichtete Arbeit in Normalenrichtung gilt im Fall einer u_z-Parametrisierung aufgrund von 5.19:

$$\frac{\mathrm{dW}_z}{\mathrm{du}_z} = u_z\frac{\mathrm{dp}_z}{\mathrm{du}_z} \overset{5.20}{=} \begin{cases} u_z\frac{1}{\mathbf{k}_z\boldsymbol{\xi}(\theta)} & \text{falls } \mathbf{u_t} \neq \mathbf{0}\,; \\ u_z K_{33}^{-1} & \text{falls } \mathbf{u_t} = \mathbf{0} \text{ und } 5.15\,; \\ u_z\frac{1}{\mathbf{k}_z\boldsymbol{\xi}(\beta)} & \text{sonst}\,. \end{cases}$$

□

Auch die u_z-/W_z-Parametrisierung erlaubt im Fall stabiler bzw. instabiler Haftung, eine numerische Integration der Differentialgleichungen zu vermeiden.

Beobachtung 11 *(Lösung der Differentialgleichungen im Haftungsfall).* Im Haftungsfall können die Differentialgleichungen aus Satz 5.13 geschlossen gelöst werden. Sei a der Wert von u_z zum Zeitpunkt des Haftungseintritts und b der Wert am Ende der jeweiligen Integrationsphase (0 am Ende der Kompressionsphase, $\overline{u}_z$ am Ende der Überbrückungsphase), so nehmen $\mathbf{u}$ und W_z am Ende der Kompressionsphase die folgenden Werte

an:

$$\mathbf{u}(b) = \begin{cases} u(a) + (b-a)\boldsymbol{e}_z & \text{falls 5.15 ;} \\ \mathbf{u}(a) + \frac{b-a}{\mathbf{k}_z \boldsymbol{\xi}_{(\beta)}} \mathbf{K}\boldsymbol{\xi}(\beta) & \text{sonst .} \end{cases} \tag{5.25}$$

$$W_z(b) = \begin{cases} W_z(a) + \frac{1}{2} K_{33}^{-1}(b^2 - a^2) & \text{falls 5.15 ;} \\ W_z(a) + \frac{b^2-a^2}{2\mathbf{k}_z \boldsymbol{\xi}_{(\beta)}} & \text{sonst .} \end{cases} \tag{5.26}$$

Satz 5.14 (Rückstellungsintegration unter W_z-Parametrisierung). *Während der durch W_z parametrisierten Rückstellungsintegration einer Kollision gilt für die Ableitung der Relativgeschwindigkeit im Kontaktpunkt:*

$$\frac{d}{dW_z}\mathbf{u} = \begin{cases} \frac{1}{u_z}\mathbf{K}\boldsymbol{\xi}(\theta) & \textit{falls } \mathbf{u_t} \neq \mathbf{0} \textit{ ;} \\ \frac{1}{K_{33}^{-1} u_z}\boldsymbol{e}_z & \textit{falls } \mathbf{u_t} = \mathbf{0} \textit{ und 5.15 ;} \\ \frac{1}{u_z}\mathbf{K}\boldsymbol{\xi}(\beta) & \textit{sonst .} \end{cases} \tag{5.27}$$

Dabei ist $\boldsymbol{e}_z$ der Einheitsvektor in Normalenrichtung und β die eindeutig bestimmte, konstante Richtung, in der die Gleitbewegung im Fall instabiler Haftung fortgesetzt wird.

Beweis. Aufgrund von Satz 5.8 können wir aus den Differentialgleichungen der allgemeinen γ-Parametrisierung 5.18 den folgenden Zusammenhang ableiten:

$$\frac{d}{dW_z}\mathbf{u} \overset{5.18}{=} \begin{cases} \mathbf{K}\boldsymbol{\xi}(\theta)\frac{dp_z}{dW_z} \overset{5.20}{=} \mathbf{K}\boldsymbol{\xi}(\theta)\frac{1}{u_z} & \text{falls } \mathbf{u_t} \neq \mathbf{0} \text{ ;} \\ \frac{1}{K_{33}^{-1}}\boldsymbol{e}_z\frac{dp_z}{dW_z} \overset{5.20}{=} \frac{1}{K_{33}^{-1} u_z}\boldsymbol{e}_z & \text{falls } \mathbf{u_t} = \mathbf{0} \text{ und 5.15 ;} \\ \mathbf{K}\boldsymbol{\xi}(\beta)\frac{dp_z}{dW_z} \overset{5.20}{=} \mathbf{K}\boldsymbol{\xi}(\beta)\frac{1}{u_z} & \text{sonst .} \end{cases}$$

□

Wir wollen an dieser Stelle noch einmal explizit darauf hinweisen, daß $\mathbf{u}$ in der durch W_z parametrisierten Rückstellungsphase differenzierbar und somit stetig ist. Die vorausgegangene Überbrückungsintegration sowie die Monotonieeigenschaften von u_z stellen sicher, daß u_z während der W_z-Integration ausschließlich positive Werte annimmt.

Analog zu den Resultaten der u_z-Parametrisierung, können auch die soeben aufgestellten Differentialgleichungen geschlossen gelöst werden. Um dies einzusehen, betrachten wir das folgende Lemma:

Lemma 5.15. *Sei f eine differenzierbare Funktion in der unabhängigen Variablen γ und sei c eine Konstante. Dann lautet die geschlossene Lösung der Differentialgleichung*

$$\frac{df}{d\gamma} = \frac{c}{f}$$

mit Anfangswert $f(a)$ an der Stelle $\gamma = b$:

$$\sqrt{f(a)^2 + 2c(b-a)}$$

Beobachtung 12 *(Lösung der Differentialgleichung im Haftungsfall).* Im Haftungsfall kann die Differentialgleichung aus Satz 5.14 geschlossen gelöst werden. Sei a der Wert von W_z zum Zeitpunkt des Haftungseintritt und b der Wert, den W_z am Ende der Rückstellungsphase annimmt, dann bestimmt sich $\mathbf{u}$ am Ende der Kollision als:

$$\mathbf{u}(b) = \begin{cases} \mathbf{e}_z \sqrt{u_z(a)^2 + \frac{2}{K_{33}^{-1}}(b-a)} & \text{falls 5.15 ;} \\ \begin{bmatrix} \sqrt{u_x(a)^2 + 2\mathbf{k}_x\xi(\beta)(b-a)} \\ \sqrt{u_y(a)^2 + 2\mathbf{k}_y\xi(\beta)(b-a)} \\ \sqrt{u_z(a)^2 + 2\mathbf{k}_z\xi(\beta)(b-a)} \end{bmatrix} & \text{sonst .} \end{cases} \tag{5.28}$$

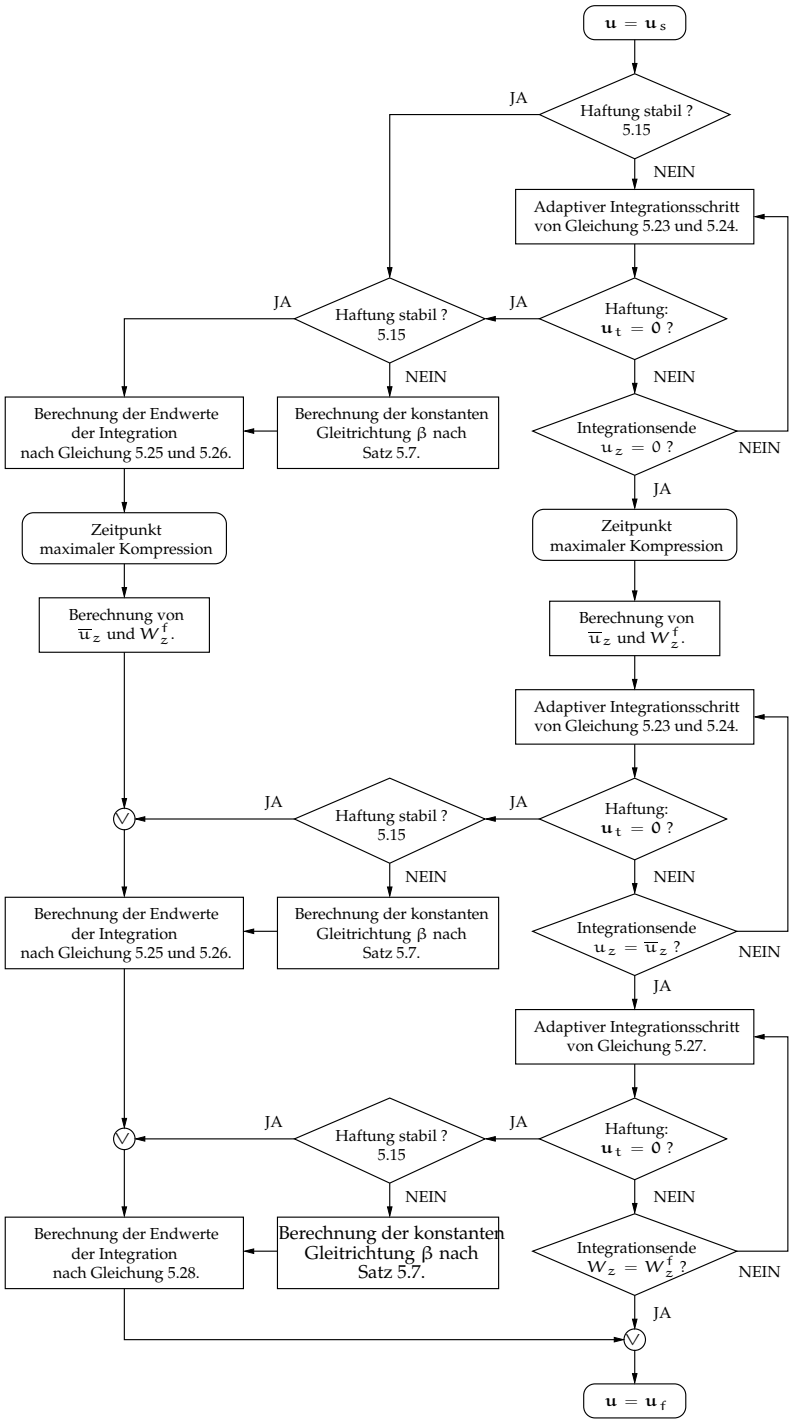
Im Fall instabiler Haftung kann man die Auswertung der Wurzelterme für u_x und u_y umgehen, indem man lediglich den Wert für u_z am Ende der Rückstellungsphase ermittelt und anschließend die Endwerte für u_x und u_y über die entsprechenden Gleichungen der u_z-Parametrisierung 5.25 ermittelt:

$$\begin{aligned} u_x &= u_x(a) + \frac{\mathbf{k}_x\xi(\beta)}{\mathbf{k}_z\xi(\beta)}\big[u_z(b) - u_z(a)\big] \\ u_y &= u_y(a) + \frac{\mathbf{k}_y\xi(\beta)}{\mathbf{k}_z\xi(\beta)}\big[u_z(b) - u_z(a)\big] . \end{aligned}$$

5.5.3 Zusammenfassung

Das Flußdiagramm aus Abbildung 5.5 veranschaulicht den Ablauf der u_z-/W_z-Integration in den beiden Kollisionsphasen. Dabei bezeichnet eine Raute eine Abfrage, die zu einer Verzweigung der Berechnung führt. Rechtecke symbolisieren einzelne durchzuführende Berechnungen und Rechtecke mit abgerundeten Ecken geben relevante Zustände an. Logische Konnektoren führen mehrere Berechnungspfade mittels einer logischen Verknüpfung zusammen. Die Kollisionsintegration verfolgt die relative Kontaktpunktgeschwindigkeit $\mathbf{u}$ über die Kompressions- und Rückstellungsphase hinweg. Die Berechnung beginnt mit dem Startwert $\mathbf{u} = \mathbf{u}_s$ und endet mit dem Wert am Ende der Kollision $\mathbf{u} = \mathbf{u}_f$. Zur Integration der Kollisionsgleichungen verwenden wir den adaptiven RUNGE-KUTTA-Differentialgleichungslöser aus [PTVF92]. Nach jedem Integrationsschritt wird die Haftbedingung $\mathbf{u}_t = \mathbf{0}$ überprüft. Haften die beiden Körper aufeinander, so ist anhand von 5.15 zwischen stabiler und instabiler Haftung zu unterscheiden. Im Fall stabiler Haftung müssen wir zunächst die eindeutige Richtung β bestimmen, in der die Gleitbewegung fortgesetzt wird (vgl. Satz 5.7). Sobald ein Haftzustand eingetreten ist, kann auf eine numerische Integration der Differentialgleichungen bis zum Ende der Impulsberechnung verzichtet werden, da die Kollisionsgleichungen geschlossen gelöst werden können. Beim Übergang von der Kompressions- in die Rückstellungsphase, d.h. zum Zeitpunkt maximaler Kompression, ist die u_z-parametrisierte Integration zu unterbrechen und der Parameterendwert der Überbrückungs- sowie Rückstellungsintegration mittels Gleichung 5.22 bzw. der Strongeschen Hypothese 5.1 zu bestimmen.

Abbildung 5.5 Flußdiagramm der Kollisionsintegration.



5.6 Energie des Systems – Verrichtete Arbeit der Kollisionsimpulse

Satz 5.16 (Energieinvariante). *Das Kollisionsmodell der impulsbasierten Simulation stellt sicher, daß das modellierte System durch eine Kollision nicht an Energie gewinnt.*

Beweis. O.B.d.A. können wir eine p_z-Parametrisierung der Kollision annehmen. Die von der Kollisionsgesamtkraft verrichtete Arbeit kann in die Beiträge der einzelnen Kraftkomponenten aufgespalten werden. Analog zur Herleitung der Gleichungen für $(dW_z/d\gamma)(\gamma)$ in Abschnitt 5.4.3 erhält man unter Berücksichtigung der Ergebnisse aus 5.12 die Ableitung der gesamten, von den Kollisionsimpulsen verrichteten Arbeit als:

$$\begin{aligned}\frac{dW}{dp_z} &= \mathbf{u}^T \frac{dp}{dp_z} \\ &= \mathbf{u}^T \boldsymbol{\xi}(\theta) \\ &= -\mu\sqrt{u_y^2 + u_y^2} + u_z \,.\end{aligned}$$

Die verrichtete Arbeit W ergibt sich somit als Integral über das Parameterintervall der Kollision:

$$W(p_{z_f}) = \underbrace{-\mu \int_0^{p_{z_f}} \sqrt{u_x^2 + u_y^2}\, dp_z}_{W_{xy}(p_{z_f}) \le 0} + \underbrace{\int_0^{p_{z_f}} u_z(p_z)\, dp_z}_{W_z(p_{z_f})} ,$$

wobei W_{xy} den Beitrag der tangential wirkenden Kräfte zur insgesamt verrichteten Arbeit bezeichnet. Da W_{xy} nichtpositiv ist, verbleibt diese Eigenschaft lediglich für die Normalenkomponente zu zeigen. Da $W_z(0) = 0$ und $u_z(p_z) \le 0$ während der Kompressionsphase $[0, p_{z_{mc}}]$ gilt, kann W_z zum Zeitpunkt maximaler Kompression nicht positiv sein.
Die Strongesche Hypothese liefert schließlich den Endwert der verrichteten Arbeit in Normalenrichtung als:

$$W_z(p_{z_f}) = (1 - e^2) \underbrace{W_z(p_{z_{mc}})}_{\le 0} \le 0 \,.$$

Somit gilt im Endzeitpunkt der Kollision:

$$W(p_{z_f}) = W_{xy}(p_{z_f}) + W_z(p_{z_f}) \le 0 \,.$$

□

Satz 5.17 (Energierhaltung). *Die Energie des Systems bleibt genau dann unverändert, wenn keine dissipativen Kräfte wirken, d.h. wenn*

(i) $e = 1$ **und** *(keine „inneren Kräfte“)*

(ii) $\mu = 0$ **oder** $\mathbf{u_t}(\gamma) = 0$ *für* $\gamma \in [\gamma_s, \gamma_f]$ *(keine „Reibungskräfte“)*

Beweis. Auch hier können wir o.B.d.A. von einer p_z-Parametrisierung der Kollision ausgehen. Im Beweis von Satz 5.16 haben wir gesehen, daß die von den Kollisionsimpulsen verrichtete Arbeit sich folgendermaßen darstellen läßt:

$$\begin{aligned} W(p_{z_f}) &= W_{xy}(p_{z_f}) + W_z(p_{z_f}) \\ &= W_{xy}(p_{z_f}) + (1-e^2)W_z(p_{z_{mc}}) \\ &= -\mu \int_0^{p_{z_f}} \sqrt{u_x^2 + u_y^2}\, dp_z + (1-e^2) \underbrace{\int_0^{p_{z_{mc}}} u_z(p_z)\, dp_z}_{<0} . \end{aligned}$$

Somit gilt:

$$W(p_{z_f}) = 0 \quad \Leftrightarrow \quad \big(\mu = 0 \ \vee \ u_x(p_z) = u_y(p_z) = 0 \text{ für } p_z \in [0, p_{z_f}]\big) \ \wedge \ e = 1 .$$

□

Auch wenn die Aussagen dieses Abschnitts dem Leser nicht sehr überraschend erscheinen mögen, so sollte man sich vor Augen führen, daß ihre Formulierung den Pionieren der impulsbasierten Simulationstechnik versagt blieben. Erst die Veröffentlichung der Arbeit von STRONGE gestattete MIRTICH 1996 [Mir96b], das *Energiekonsistenzproblem* zu lösen. Die Sätze dieses Abschnitts tragen somit ganz wesentlich zur Plausibilität des Kollisionsmodells bei.

In diesem Sinne ist auch der nun folgende Satz zu werten, der eine Aussage über die Weg(un)abhängigkeit der gesamten, von den Kollisionsimpulsen an den beiden Körpern verrichteten Arbeit trifft. Bisher haben wir uns primär mit der in Normalenrichtung verrichteten Arbeit beschäftigt, da diese im Zusammenhang mit der Berechnung der Integrationsgrenzen eine wichtige Rolle spielt. Der Wert von W_z am Ende der Kompressionsphase mußte dabei mittels numerischer Integration berechnet werden. Auch die gesamte verrichtete Arbeit kann analog zu ihrem Beitrag in Normalenrichtung als Integral über das zu betrachtende Parameterintervall $[\gamma_0, \gamma_1]$ dargestellt werden:

$$\begin{aligned} \Delta W(\gamma_1) &= \int_{\gamma_0}^{\gamma_1} \mathbf{u}(\gamma)^T \frac{d}{d\gamma} p(\gamma)\, d\gamma \\ &\overset{5.11}{=} \int_{\gamma_0}^{\gamma 1} \mathbf{u}(\gamma)^T \mathbf{K}^{-1} \frac{d}{d\gamma} \mathbf{u}(\gamma)\, d\gamma . \end{aligned} \tag{5.29}$$

Diese Gleichung suggeriert, daß man zur Bestimmung von W die Ableitung $(dW/d\gamma)$ über das Parameterintervall integrieren muß. Wir werden im folgenden jedoch zeigen, daß bereits die Kenntnis von $\mathbf{u}(\gamma_s)$ sowie $\mathbf{u}(\gamma_f)$ zur Berechnung von W genügt, d.h. daß W *wegunabhängig* ist.

Satz 5.18 (Wegunabhängigkeit von W). *Betrachten wir ein beliebiges Parameterintervall $[\gamma_0, \gamma_1]$ während der Kollision. Sei $\mathbf{u}(\gamma_0)$ und $\mathbf{u}(\gamma_1)$ die relative Kontaktpunktgeschwindigkeit am Anfang und am Ende dieses Intervalls. Dann ist die gesamte, von den Kollisionsimpulsen verrichtete Arbeit unabhängig von dem Weg, auf dem die Arbeit verrichtet wurde.*

$$\Delta W(\gamma_1) = \frac{1}{2}\big[\mathbf{u}(\gamma_1) + \mathbf{u}(\gamma_0)\big]^T \mathbf{K}^{-1} \Delta \mathbf{u}(\gamma_1) ,$$

mit $\Delta W(\gamma_1) := W(\gamma_1) - W(\gamma_0)$ *und* $\Delta \mathbf{u}(\gamma_1) := \mathbf{u}(\gamma_1) - \mathbf{u}(\gamma_0)$.

Beweis. Aufgrund unserer Überlegungen in 5.29 wissen wir:

$$\begin{aligned}\frac{dW}{d\gamma} &= u(\gamma)^T \mathbf{K}^{-1} \frac{d}{d\gamma} u(\gamma) \\ &= \frac{1}{2}\left[u(\gamma)^T \mathbf{K}^{-1} \frac{d}{d\gamma} u(\gamma) + u(\gamma)^T \mathbf{K}^{-1} \frac{d}{d\gamma} u(\gamma)\right] .\end{aligned}$$

Aufgrund der Symmetrie von $\mathbf{K}^{-1}$ gilt:

$$\begin{aligned}\frac{dW}{d\gamma} &= \frac{1}{2}\left[\left(\frac{d}{d\gamma}\mathbf{u}(\gamma)\right)^T \mathbf{K}^{-1}\mathbf{u}(\gamma) + \mathbf{u}(\gamma)^T \mathbf{K}^{-1} \frac{d}{d\gamma} u(\gamma)\right] \\ &= \frac{1}{2}\frac{d}{d\gamma}\left[\mathbf{u}(\gamma)^T \mathbf{K}^{-1}\mathbf{u}(\gamma)\right] .\end{aligned}$$

Integriert man diese Gleichung über dem Parameterintervall $[\gamma_0, \gamma_1]$, so erhält man:

$$\begin{aligned}&\Delta W(\gamma_1) \\ &= \frac{1}{2}\left[\mathbf{u}(\gamma_1)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_1) - \mathbf{u}(\gamma_0)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_0)\right] \\ &= \frac{1}{2}\left[\mathbf{u}(\gamma_1)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_1) - \mathbf{u}(\gamma_1)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_0) + \mathbf{u}(\gamma_1)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_0) - \mathbf{u}(\gamma_0)^T \mathbf{K}^{-1}\mathbf{u}(\gamma_0)\right] \\ &= \frac{1}{2}\left[\mathbf{u}(\gamma_1)^T \mathbf{K}^{-1}\Delta\mathbf{u}(\gamma_1) + \mathbf{u}(\gamma_0)^T \mathbf{K}^{-1}\Delta\mathbf{u}(\gamma_1)\right] \\ &= \frac{1}{2}\left[\mathbf{u}(\gamma_1) + \mathbf{u}(\gamma_0)\right]^T \mathbf{K}^{-1}\Delta\mathbf{u}(\gamma_1) .\end{aligned}$$

□

Diese Aussage wird uns im folgenden Abschnitt, welcher der Analyse einer speziellen Klasse von Kollisionen gewidmet ist, von Nutzen sein.

5.7 Permanenter Kontakt und Mikrokollisionen

Das Kontaktmodell der impulsbasierten Methode ist sehr gut geeignet, Systeme zu simulieren, in denen die Beschreibung der Kontaktbeziehungen zwischen den Körpern anhand von makroskopischen Zwangsbedingungen schwierig oder gar nicht möglich ist. Solange die Körper nicht in *permanentem Kontakt* miteinander stehen, d.h. aufeinander rollen, gleiten oder gar liegen, erscheint die mikroskopische Betrachtungsweise des impulsbasierten Ansatzes als eine sehr natürliche Sicht auf das tatsächliche Kontaktverhalten des zu simulierenden Systems.

Die Kollisionsauflösung, basierend auf dem in diesem Kapitel beschriebenen Modell, ist nicht bemüht, Zwangsbedingungen, wie sie bei Formen permanenten Kontakts gelten, zu erkennen und für eine einfache Beschreibung des Kontaktverhaltens zu nutzen. Dahinter steht die Überzeugung, daß die isolierte und einheitliche Behandlung

von Kollisionen anhand der lokalen Daten des Kontaktpunkts korrektes makroskopisches Verhalten erzwingt. Aus diesem Grunde wird in der impulsbasierten Methode permanenter Kontakt als eine zeitlich dichtgedrängte Folge schwacher Kollisionen modelliert. Diese Vorgehensweise kann man daher als eine Approximation des tatsächlichen physikalischen Verhaltens auffassen. Um die Plausibilität des Modells zu verdeutlichen, wollen wir die impulsbasierte Behandlung der erwähnten Kontaktformen etwas genauer erläutern.

Betrachten wir zunächst folgende Situation. Ein Buch liegt auf einem Tisch, modelliert durch eine Folge vieler schwacher Kollisionen zwischen Tisch und Ecken des Buches. Durch die Kollisionen verliert das Buch permanent an Energie, so daß es immer tiefer in die ε_c-Hülle des Tisches hineinsinkt. Dies bedeutet, daß der Abstand und damit die Kollisionszeitschätzungen gegen Null streben und das Simulationssystem unter dem Berechnungsaufwand einbricht. Um dieses Verhalten in der Praxis zu umgehen, schlägt MIRTICH in [Mir96b] eine Klassifizierung von Kollisionen vor. Kollisionen, deren anfängliche Relativgeschwindigkeit in Kollisionsrichtung „sehr klein“ist, sollten in einer von unserer bisherigen Vorgehensweise abweichenden Weise behandelt werden. Dabei darf jedoch das *Paradigma der Lokalität*, d.h. der ausschließlichen Verwendung lokaler Informationen, nicht verletzt werden.

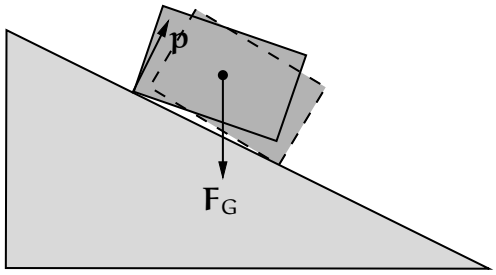
Definition 5.4. Eine Kollision heißt *Mikrokollision*, wenn die folgende Bedingung erfüllt ist:

$$-u_z(\gamma_s) < \sqrt{2g\varepsilon_c}\,,$$

wobei g die Erdbeschleunigung bezeichnet. □

Die Definition wird von folgender Anschauung getragen. Liegt permanenter Kontakt zwischen zwei Körpern vor, dann ist die Normalenkomponente der relativen Kontaktpunktgeschwindigkeit sehr klein. Wir bezeichnen daher nur solche Kollisionen als Mikrokollisionen, für die gilt, daß $|u_z|$ zum Zeitpunkt des Kollisionseintritts kleiner als die Geschwindigkeit eines Körpers ist, der aus der Ruhelage in freiem Fall durch die Kollisionshülle (ε_c-Hülle) des Kontaktpartners hindurch fällt.
Bei der Auflösung dieser Mikrokollisionen wollen wir den Rückstellungskoeffizient e künstlich erhöhen, so daß die Kontaktpartner sich nach der Kollision ausreichend weit voneinander entfernen. Durch diese Vorgehensweise wird in der Rückstellungsphase mehr Energie an die Körper zurückgegeben, als in der Kompressionsphase in ihnen gespeichert wurde. Unglücklicherweise bedeutet dies, daß die in Satz 5.16 garantierte Energiekonsistenz unseres Modells nicht mehr gewährleistet ist. Das System gewinnt durch Mikrokollisionen an Energie, sobald der künstlich erhöhte Rückstellungskoeffizient den Wert 1 überschreitet. Jedoch ist dieser Energiegewinn aufgrund der geringen Kontaktgeschwindigkeit vernachlässigbar gering.
Wir suchen also eine Funktion e, die der Durchdringungstiefe der Kollisionshülle $d = \varepsilon_c - \delta(\mathcal{K}_1, \mathcal{K}_2) \in [0, \varepsilon_c]$ einen Rückstellungskoeffizienten $e(d) \in [e_{def}, e_{max}]$ zuordnet, wobei e_{def} der ursprüngliche Wert und e_{max} eine Obergrenze für den Rückstellungskoeffizienten bezeichnet. Wenn die Durchdringung sehr gering ist, dann sollte $e(d)$ in der Nähe von e_{def} liegen; falls d sich jedoch dem Maximalwert ε_c nähert, so

Abbildung 5.6 Unabhängig von der Stärke der Reibungskraft bewirken die ballistischen Bewegungsphasen nach der impulsbasierten Kollisionsauflösung eine Gleitbewegung des Quaders auf der Rampe.



muß $e(c)$ entsprechend groß werden. Wir haben in unserer Implementierung e als eine Funktion 2. Grades mit Scheitelpunkt in $d = 0$ und einem maximalen Funktionswert $e_{max} = 5.0$ gewählt:

$$\begin{aligned} e : [0, \varepsilon_c] \subseteq \mathbb{R}^+ &\longrightarrow [e_{def}, e_{max}] \subseteq \mathbb{R}^+ \\ e(d) &\longmapsto \frac{e_{max} - e_{def}}{\varepsilon_c^2} d + e_{def} \end{aligned}$$

Diese Vorgehensweise bewirkt eine Vibrationsbewegung des Buches auf dem Tisch, wobei die Amplitude dieser Bewegung in der Größenordung von ε_c und somit außerhalb des sichtbaren Bereichs liegt. Allerdings, und das wollen wir hier deutlich aufzeigen, stellt diese Vorgehensweise einen *ad-hoc-Ansatz* dar, der das geschilderte Problem nicht endgültig löst. Ein geeignet dimensionierter Bücherstapel würde ab einer bestimmten Größe das Simulationssystem aufgrund permanenter Impulspropagation in die Knie zwingen.

Ein weiteres Problem kann man sich an der Bewegung eines Quaders auf einer schiefen Ebene verdeutlichen (Abbildung 5.6). Solange die Reibungskraft nicht stark genug ist, um die Gleitbewegung des Quaders zu stoppen, liefert unsere Simulation qualitativ das richtige Verhalten. Dominiert die Reibung die Wirkung der Gravitation, so würde der Quader dennoch die Ebene herabgleiten, obwohl in jeder Kollision die Gleitbewegung durch Haftreibung unmittelbar zum Stillstand gebracht wird. Ursächlich für dieses Verhalten sind ballistische Bewegungsphasen, denen der Quader nach jeder noch so schwachen Kollision in unserem Modell unterliegt. In diesen Phasen wirkt ausschließlich die Gravitationskraft, die den Quader für ein kurzes Zeitintervall die Ebene hinab beschleunigt. Glücklicherweise können wir Probleme dieser Art nicht nur sehr einfach und auf lokalen Informationen basierend erkennen, sondern auch elegant und physikalisch plausibel lösen. Dazu überprüfen wir im Fall einer Mikrokollision, ob der Impuls $\mathbf{p}_r$, der die Kontaktpunktgeschwindigkeit umkehrt, im Reibungskegel 5.15 liegt. $\mathbf{p}_r$ berechnet sich daher folgendermaßen:

$$\mathbf{p}_r = \mathbf{K}^{-1} \Delta u(\gamma_f) = \mathbf{K}^{-1} \big(-2\mathbf{u}(\gamma_s) \big) = -2\mathbf{K}^{-1}\mathbf{u}(\gamma_s) \, .$$

Ist die Reibungskraft stark genug, so daß stabile Haftung während der gesamte Kollisionsdauer vorliegt, dann wenden wir den geschwindigkeitsinvertierenden Impuls auf die Objekte an. Diese Vorgehensweise hat nicht nur den Vorteil, daß die Berechnungskosten deutlich unter denen der „Standard"-Kollisionsauflösung liegen, da keine Integration der Differentialgleichungen erforderlich ist; sie ist außerdem physikalisch plausibel, weil sie sicherstellt, daß der ermittelte Kollisionsimpuls keine Arbeit an den beteiligten Körpern verrichtet. Diese Aussage folgt unmittelbar aus Satz 5.18 über die Wegunabhängigkeit von W. Ist Haftbedingung nicht erfüllt, ermitteln wir den Impuls über das Berechnungsverfahren, das wir bereits für Mikrokollisionen vorgestellt haben.

Mikrokollisionen sind ein ad-hoc-Ansatz, um die impulsbasierte Simulationsmethode auch im Fall permanenten Kontakts praktikabel zu machen. Die Ergebnisse des vorangegangenen Abschnitts, die wichtige Aussagen über die Plausibilität des Modells treffen, können unter Berücksichtigung des ersten Typs von Mikrokollisionen nicht aufrechterhalten werden. Gerade am Beispiel des Buchs auf dem Tisch wird die eingeschränkte Verwendbarkeit des Ansatzes und die Notwendigkeit des Übergangs zu zwangsbasierten Verfahren im Fall permanenten Kontakts offensichtlich.

5.8 Zusammenfassung

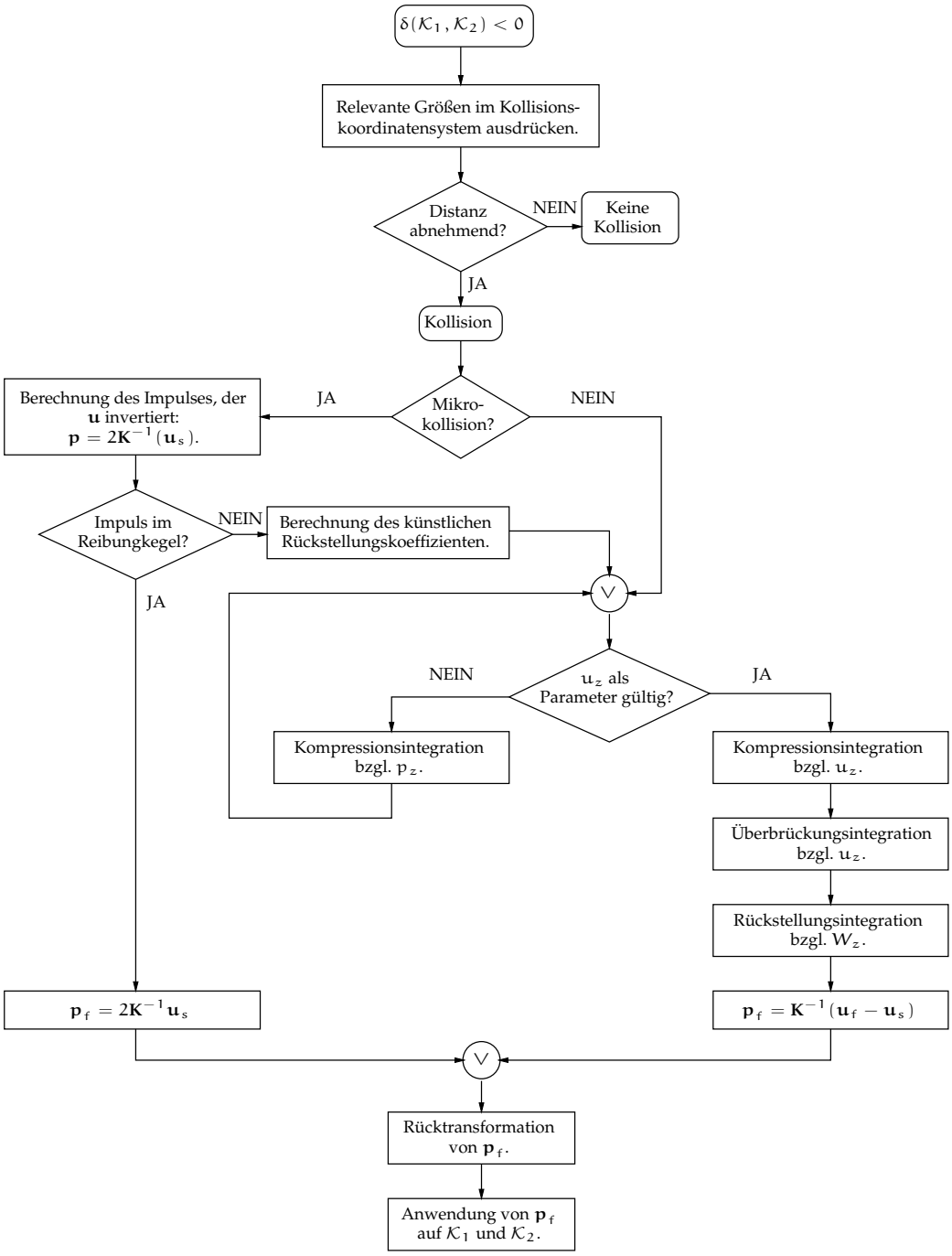
Um den Ablauf der Kollisionsauflösung zu verdeutlichen, wollen wir die durchzuführenden Berechnungen entsprechend der zu treffenden Fallunterscheidungen in einem Flußdiagramm darstellen (Abbildung 5.7). Die Kollisionsauflösung wird aufgerufen, sobald die Kollisionserkennung als Abstand zweier Körper $\mathcal{K}_1$, $\mathcal{K}_2$ einen Wert $\delta(\mathcal{K}_1, \mathcal{K}_2) < \varepsilon_c$ ermittelt hat. Ausgehend von dem vorliegenden Kontakttyp, d.h. Kante-Kante- bzw. Punkt-Fläche-Kontakt, wird das Kollisionskoordinatensystem (vgl. Abschnitt 5.2) bestimmt. Anschließend werden alle relevanten Dynamikdaten in diesem Bezugssystem ausgedrückt. Ein von der Kollisionserkennung gemeldeter Kontakt ist nur dann aufzulösen, wenn die Distanz der Objekte in Kontaktrichtung abnimmt. Diese Entscheidung wird anhand der Abfrage $u_z < 0$ (vgl. Abschnitt 5.2) getroffen.

Die relative Kontaktpunktgeschwindigkeit $\mathbf{u}$ wird vor der Integration gesichert ($\mathbf{u}_s$), da der übertragene Impuls $\mathbf{p}$ aus der Veränderung von $\mathbf{u}$ berechnet wird:

$$\mathbf{p} = \mathbf{K}^{-1}(\mathbf{u} - \mathbf{u}_s) \,. \tag{5.30}$$

Im ersten Schritt ist die Kollision entsprechend unseren Überlegungen des vorangegangenen Abschnittes zu klassifizieren. Im Fall einer Mikrokollision , d.h. falls $u_z < \sqrt{2g\varepsilon_c}$ gilt, berechnen wir den Impuls, der $\mathbf{u}_s$ invertiert: $p_f = \mathbf{K}^{-1}\mathbf{u}_s$. Falls dieser Impuls im Reibungskegel liegt, liefern wir $\mathbf{p}_f$ als Kollisionsimpuls zurück. Andernfalls berechnen wir den künstlichen Rückstellungskoeffizienten e, wie in Abschnitt 5.7 beschrieben, und führen die „Standard"-Kollisionsintegration (vgl. Abschnitt 5.5.3) durch. Die „Standard"-Kollisionsintegration verfolgt die relative Kontaktpunktgeschwindigkeit $\mathbf{u}$ durch Integration der Kollisionsgleichungen über die Kompressions- und Rückstellungsphase hinweg. Im allgemeinen wird die Größe u_z als Integrationsparameter für

Abbildung 5.7 Flußdiagramm des Kollisionsauflösungsverfahrens.



die Kompressionsphase gewählt. Um die Eignung von u_z zu verifizieren, ist die Bedingung $\frac{du_z}{dp_z} > 0$ zu überprüfen (vgl. Abschnitt 5.5.2). Falls diese notwendige Voraussetzung für den Einsatz von u_z verletzt ist, führen wir zunächst die Kompressionsintegration bezüglich dem Parameter p_z durch (vgl. Abschnitt 5.5.1), und zwar solange, bis u_z das Gültigkeitskriterium erfüllt. Ist schließlich u_z als Parameter zulässig, so setzen wir die Verfolgung der relativen Kontaktpunktgeschwindigkeit anhand der in Abschnitt 5.5.2 beschriebenen Integration von u_x, u_y und W_z in der Kompressionsphase , sowie von $\mathbf{u}$ in der Rückstellungsphase fort.

Zum Zeitpunkt maximaler Kompression ($u_z = 0$) wird die Kompressionsintegration unterbrochen, um die Parameterendwerte $\overline{u}_z$ bzw. W_z^f der Überbrückungs- und Rückstellungsintegration zu bestimmen. Die Berechnung liefert schließlich die relative Kontaktpunktgeschwindigkeit $\mathbf{u}_f$ am Ende der Kollision. Der Kollisionsimpuls $\mathbf{p}_f$ kann nun anhand von Gleichung 5.30 bestimmt werden. Anschließend wird $\mathbf{p}_f$ aus dem Kollisionskoordinatensystem in das Weltsystem zurücktransformiert und auf die an der Kollision beteiligten Körper angewendet:

$$\begin{aligned} \Delta \mathbf{v}_1 &= \frac{1}{m_1}\mathbf{p}_f\,, & \Delta \boldsymbol{\omega}_1 &= \mathbf{I}_1^{-1}(\mathbf{r}_1 \times \mathbf{p}_f)\,, \\ \Delta \mathbf{v}_2 &= -\frac{1}{m_2}\mathbf{p}_f\,, & \Delta \boldsymbol{\omega}_2 &= -\mathbf{I}_2^{-1}(\mathbf{r}_2 \times \mathbf{p}_f)\,. \end{aligned}$$

Evaluation des Simulationsverfahrens und Integration in die Softwarebibliothek SiLVIA

6 Softwareumgebung und Beispielsimulationen

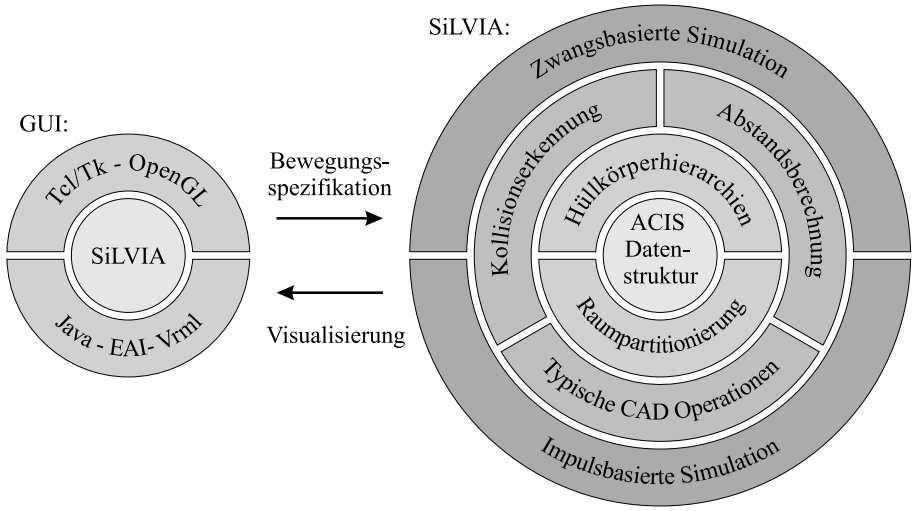
6.1 Die Softwarebibliothek SiLVIA

Die in dieser Arbeit vorgestellten Datenstrukturen und Algorithmen wurden im Rahmen des SILVIA-Projektes am Lehrstuhl von Professor Hotz an der Universität des Saarlandes implementiert. Sie sind Bestandteil der Software-Bibliothek SILVIA (Simulation Library for Virtual Reality and Interactive Applications), einer in C++ realisierten Plattform zur Durchführung von Kollisions- und Kontaktberechnungen unter Echtzeitanforderungen [HKL+99]. Der Schwerpunkt der Entwicklung liegt dabei auf der Bereitstellung effizienter und physikalisch realitätsgetreuer Verfahren zur Echtzeitsimulation des dynamischen Verhaltens kollidierender starrer Körper. Folglich liegen die Einsatzgebiete der Bibliothek im Bereich (Distributed) Virtual Reality, in der *Realitätstreue* der simulierten Abläufe *und Echtzeitanforderungen* in Einklang gebracht werden müssen. Die bereitgestellten Routinen erlauben beispielsweise die Entwicklung von *Virtual Prototyping*-Anwendungen, welche als Ergänzung bestehender CAD-Systeme eine rechnergestützte Designerprobung ermöglichen.

Auch wenn die Simulationsverfahren im Mittelpunkt des Forschungsinteresses lagen, mußte zunächst ein (objektorientiertes) *Modell der virtuellen Welt* erstellt und ein *graphisches Benutzerinterface* (GUI) zur Visualisierung der Simulationsdaten sowie zur Eingabe der (verteilten) Benutzerwünsche entwickelt werden. Die Verteilung der virtuellen Welt ist durch ein einfaches *Client-Server-Konzept* [Kle99] realisiert, in dem die Klienten lediglich Visualisierungsaufgaben übernehmen, während der Server die zentralen Simulationsberechnungen durchführt. Die Benutzerschnittstelle wurde dabei bewußt von der Simulationsbibliothek getrennt, wie Abbildung 6.1 verdeutlicht. Die Kommunikation zwischen beiden Modulen beschränkt sich auf das Senden von Visualisierungsdaten an das GUI bzw. die Übermittlung von Bewegungswünschen in umgekehrter Richtung. Diese strikte Trennung wurde vorgenommen, um die gewünschte *Plattformunabhängigkeit* sicherzustellen. Die Abstraktion von einer konkreten Benutzerschnittstelle ermöglicht beispielsweise die alternative Verwendung eines OpenGL®-Tcl/Tk®GUIs oder eines VRML-Browsers zur Spezifikation der Bewegungswünsche und zur Visualisierung der Simulationsdaten. Die Integration des GUIs in eine HTML-Seite genügt somit, um eine verteilte virtuelle Welt zu schaffen, in der mehrere Benutzer kollaborativ Objekte manipulieren können.

Die schalenförmige Architektur der eigentlichen Simulationsbibliothek ist in Ab-

Abbildung 6.1 Der schalenförmige Aufbau der Simulationsbibliothek SILVIA.



bildung 6.1 dargestellt. Den Kern von SILVIA bildet eine Datenstruktur zur randorientierten Körperrepräsentation. Diese folgt der ACIS®-Spezifikation [Spa], die als wohldokumentierter Industriestandard den Kern vieler kommerzieller CAD-Systeme darstellt. Die *ACIS®-Datenstruktur* ermöglicht auf diese Weise die Entwicklung einer Vielzahl von Import- und Exportfiltern und somit einen reibungslosen Datenaustausch zwischen CAD-System und Simulationswerkzeug. Die Umsetzung der ACIS®-Spezifikation in SILVIA beschränkt sich derzeit auf die zur geometrisch-topologischen Repräsentation von Polyedern relevanten Komponenten. Eine Erweiterung bezüglich gekrümmter Oberflächen ist jedoch ohne Schwierigkeiten möglich.

Auf dieser zentralen Datenstruktur bauen zwei wesentlich Konzepte dieser Arbeit auf, nämlich der Hüllkörperbaum zur Verwaltung einer *Hierarchie von Hüllkörpern* (vgl. Abschnitt 3.5 und 3.6) und eine *hierarchische Raumpartitionierung* (Abschnitt 4.4.6) zur effizienten Lokalisation von Objekten in der virtuellen Welt. Wie wir in Abschnitt 2.6.3 bereits dargelegt haben, sind beide Techniken unumgänglich, will man Szenarien mit vielen Körpern komplexer Geometrie effizient handhaben. Die Einsatzgebiete beider Datenstrukturen innerhalb der SILVIA-Bibliothek gehen jedoch weit über die in dieser Arbeit vorgestellten Konzepte hinaus. So wird die Raumpartitionierung nicht nur zur Bestimmung räumlich naher Bewegungshüllkörper verwendet, sondern identifiziert zusätzlich als Vorverarbeitungsschritt der Kollisionserkennung (broad-phase) die Paare sich potentiell überlappender Objekte. Auf diese Weise wird der Aufwand der statischen Kollisionserkennungs- bzw. Abstandstests entscheidend reduziert. Die Hüllkörperhierarchie wird sowohl zur Beschleunigung der Abstandsberechnung (vgl. Kapitel 3) als auch zur Realisierung einer effizienten dynamischen Kollisionserkennung auf der Basis von [Eck98] eingesetzt. Beide Verfahren beruhen auf Divide-and-Conquer-Strategien, wobei das Kollisionserkennungsverfahren auf den in Abschnitt 2.6.3 vor-

gestellten hüllkörperbasierenden Ausschlußtest für Teilbäume der Hierarchie zurückgreift. Daneben unterstützen Hüllkörperhierarchien *typische CAD-Operationen* (z.B. Schnittbildung), die sich im Rahmen der Objektkonstruktion/-modifikation als hilfreich erweisen [Beh99]. Die Funktionalität der SILVIA-Bibliothek umfaßt auf diesem Gebiet Routinen zur Translation und Rotation von Objekten, zur Objektgenerierung mittels Extrusion und Rotation, sowie zur Bestimmung physikalischer Größen wie Masse, Schwerpunkt und Trägheitsmatrizen. Um die Grundlage für die geplante Unterstützung von Objekten mit gekrümmten Oberflächen zu legen, wurden Algorithmen zu deren Approximation mittels Dreiecksflächen entwickelt [Sch99]. Eine solche Oberflächentriangulierung ist derzeit zwingend erforderlich, da bestimmte Algorithmen (z.B. schnelles Rendering) eine polygonale Randrepräsentation voraussetzen.

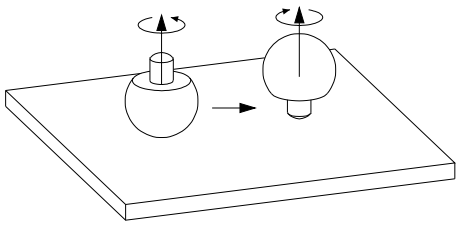
Die von der SILVIA-Bibliothek bereitgestellte *Abstandsberechnung* entspricht dem in Kapitel 3 entwickelten Branch-and-Bound-Verfahren und liefert den Euklidischen Abstand sowie ein Paar nahester Punkte als Zeugen des Abstandsminimums. Mit Hilfe des Kollisionsheaps (Abschnitt 4.3) wird die Abstandsberechnung zu einer dynamischen Kollisionserkennung erweitert, deren Eignung jedoch auf Simulationen beschränkt ist, in denen keine online-Bewegungsspezifikationen erfolgen. Ist dem Benutzer dagegen eine Manipulation der Objekte während der Simulation gestattet, so kommt das in [Eck98] vorgestellte dynamische Kollisionserkennungsverfahren zum Einsatz.

Auf der Grundlage dieser Basisalgorithmen wurden die Verfahren zur Dynamiksimulation starrer Körper entwickelt. Die komplementären Vorzüge und Nachteile der impuls- und zwangsbasierten Techniken (vgl. Abschnitt 2.2.3) legten die Integration von algorithmischen Vertretern beider Simulationsparadigmen nahe. Das *impulsbasierte Simulationsverfahren* ist dabei die Umsetzung des in Kapitel 5 vorgestellten Konzeptes von MIRTICH [Mir96b]. Die *zwangsbasierten Algorithmen* gehören ausnahmslos der Klasse der sogenannten „analytischen“ Verfahren an, was bereits durch die avisierten Einsatzgebiete der Bibliothek erzwungen wird. Es handelt sich dabei um die in Abschnitt 2.2.1 vorgestellten Eigenentwicklungen, welche in [BS98b, BS98a, SS98a, SS98b] detailliert beschrieben werden.

6.2 Die Simulation „mechanischer Spielzeuge“

Das in Abschnitt 5 vorgestellte Kollisionsauflösungsverfahren erweist sich in der Praxis als schwierig zu implementieren. Numerisch schlecht konditionierte Daten und degenerierte Kontaktsituationen sind in realen Simulationen nicht selten und erfordern neben dem Einsatz numerisch robuster Algorithmen zur Matrixinvertierung sowie zur Integration von Differentialgleichungen die Einführung von Toleranzen und weiteren Fallunterscheidungen. Als besonders kritische Phase der Implementierung stellt sich die Testphase der Kollisionsauflösung dar. Während man reibungsfreie Stöße geometrisch einfacher Körper wie beispielsweise Kugeln anhand der Bewegungsbahnen und der zeitlichen Geschwindigkeitsverläufe noch nachvollziehen kann, ist der Einfluß von Reibungskräften oder komplexen Geometrien auf die Simulationsdaten nur sehr sel-

Abbildung 6.2 Der Steh-Auf-Kreisel.



ten vorauszusagen. Aus diesem Grund haben wir uns nach bekannten Experimenten der klassischen Mechanik umgesehen, die einen besonders charakteristischen oder gar kuriosen Verlauf besitzen und ausschließlich unter Berücksichtigung von Reibung analysiert und erklärt werden können. Vor diesem Hintergrund erschienen die folgenden „mechanischen Spielzeuge“ als besonders geeignet:

- der klassische Steh-Auf-Kreisel [Bra52, Hug52, Syn52, Pli54, Cam55, Coh77, Or94],
- die Münze mit Loch,
- das drehende Ellipsoid,
- der keltische Wackelstein [GH86].

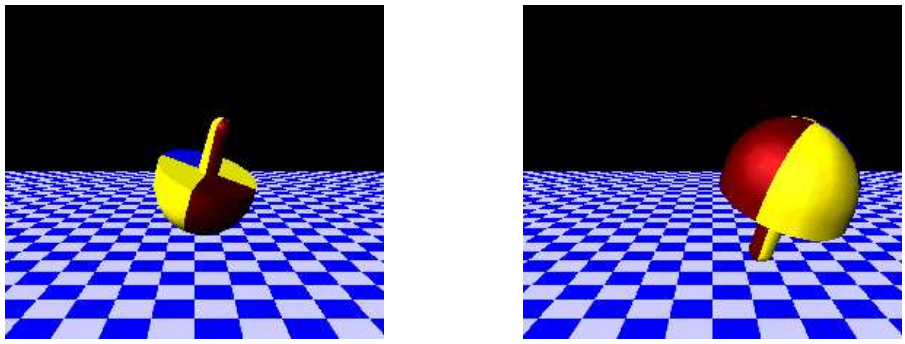
Das primäre Ziel, das wir mit der Simulation dieser Spielzeuge verfolgten, bestand darin, deren charakteristisches Verhalten *qualitativ* nachvollziehen zu können. Dies ermöglichte uns, Fehler in der Implementierung zu identifizieren und die *Güte und Anwendbarkeit des Reibungsmodells* zu beurteilen. Die Kollisionserkennung spielte aufgrund der einfachen Geometrie und der Beschränkung auf zwei Objekte in den hier vorgestellten Beispielsimulationen lediglich eine untergeordnete Rolle. Anschließend haben wir die Simulationsdaten *quantitativ* mit dem zwangsbasierten Dynamiksimulationsverfahren von SAUER [SS98a, SS98b] verglichen, was zu verblüffenden Ergebnissen [SSL98] geführt hat.

Wir wollen im folgenden die einzelnen Experimente beschreiben und das qualitativ realitätsgetreue Simulationsverhalten anhand von Simulationsdaten und -schnappschüssen belegen.

6.2.1 Der Steh-Auf-Kreisel

Das erste Experiment betrifft den klassischen *Steh-Auf-Kreisel*, der in Abbildung 6.2 dargestellt ist. Dreht man den Kreisel langsam auf einer ebenen Fläche, so bleibt sein dynamisches Verhalten stabil. Aufgrund der Verschiebung des Massenschwerpunktes gegenüber der Situation im Fall der perfekten Kugel wandert der Stiel nach unten und setzt eventuell auf der Unterlage auf (Zweipunktkontakt). Dreht man den Kreisel

Abbildung 6.3 Zwei Momentaufnahmen der Kreiselsimulation.



schneller an, so daß eine bestimmte Winkelgeschwindigkeit überschritten wird, wobei die Reibungskraft zwischen Kreisel und Auflagefläche ausreichend groß sein muß, dann stellt sich der Kreisel auf den Stiel und dreht auf diesem solange weiter, bis der stetige Energieverlust ihn umfallen läßt.

Die Analyse sowie die Erklärung dieses Verhaltens ist komplex und kann in der Literatur [Bra52, Hug52, Syn52, Pli54, Cam55, Coh77, Or94] nachgelesen werden. Es ist jedoch erwiesen, daß Reibung für den zu beobachtenden Effekt verantwortlich ist. Abbildung 6.3 zeigt das Verhalten des Kreisels anhand von Momentsaufnahmen der impulsbasierten Simulation.

6.2.2 Die Münze mit Loch

Ein weiterer Effekt, den wir rechnergestützt nachvollziehen konnten, läßt sich bei der Drehung einer *Münze mit ausgestanztem Loch* auf einer ebenen Fläche beobachten. Bohrt man in eine Münze ein Loch außerhalb ihres Mittelpunktes und dreht die Münze schnell genug, wobei das Loch zu Beginn der Drehung oberhalb des Mittelpunktes liegt, so wird sich die Aussparung im Laufe des Experimentes nach unten bewegen. Die Ursache dieses Effektes liegt, ähnlich wie im Fall des Steh-Auf-Kreisels, in der Kombination aus Exzentrität des Massenschwerpunktes und der Reibung zwischen Münze und Auflagefläche. Ausgangs- und Endlage der Münze sind in Abbildung 6.4 dargestellt. Beide Abbildungen wurden aus den Simulationsdaten gewonnen.

6.2.3 Das drehende Ellipsoid

Das Experiment des *drehenden Ellipsoids* kann der Leser auf einfache Weise zu Hause nachvollziehen. Man benötigt lediglich ein hartgekochtes Ei, das man auf einer ebenen Unterlage wie in Abbildung 6.5 dargestellt dreht. Das Ei stellt sich im Laufe der Objektbewegung auf und rotiert dabei um die längste Symmetrieachse. Das Simulationsergebnis ist in Abbildung 6.5 dargestellt.

Abbildung 6.4 Die Simulation einer drehenden Münze mit ausgestanztem Loch.

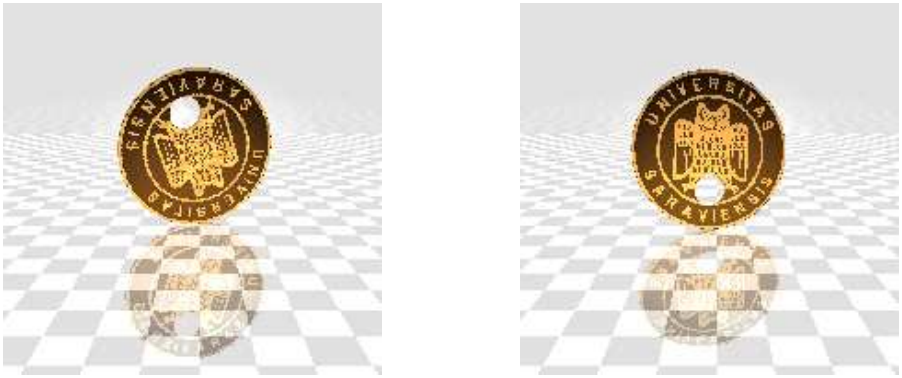


Abbildung 6.5 Das drehende Ellipsoid.

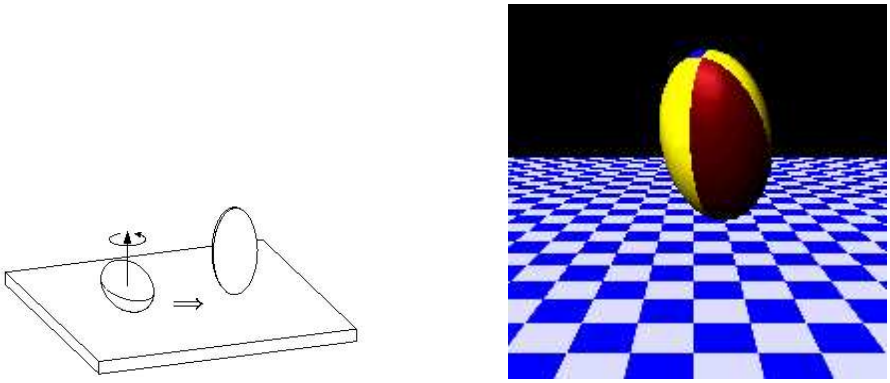
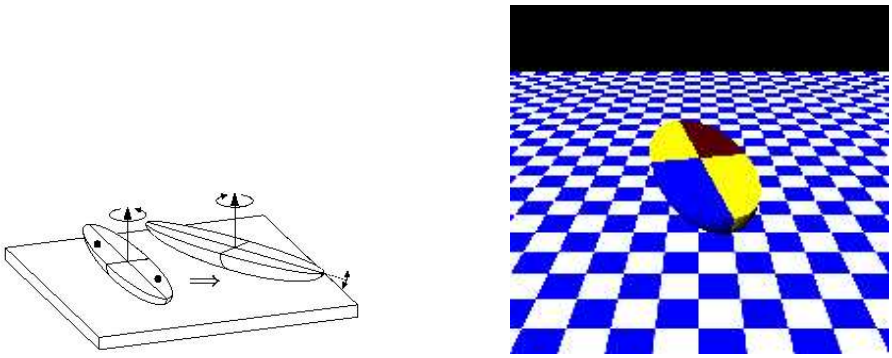


Abbildung 6.6 Der keltische Wackelstein.



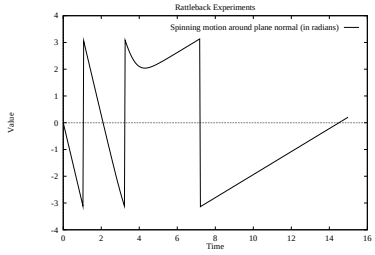
6.2.4 Der keltische Wackelstein

Außerordentlich verblüffend ist das Verhalten des *keltischen Wackelsteines* [GH86]. Dabei handelt es sich um die Hälfte eines Ellipsoids, das entlang der längsten Symmetrieachse durchgeschnitten wurde. Aus physikalischer Sichtweise handelt es sich um einen Kreisel. Die Eigentümlichkeit des Kreisels besteht nun darin, daß er nicht rotationssymmetrisch zu seiner Figurenachse ist. Diese Eigenschaft erzielt man beispielsweise, indem man die Ellipsoidhälften imaginär viertelt, wie in Abbildung 6.6 angedeutet, und nur die beiden Paare schräg gegenüberliegender Viertel mit gleicher Dichte ausstattet. In der Abbildung unterscheiden sich also schwarze und weiße Viertel in ihrer Dichte, während die beiden schwarzen bzw. die beiden weißen Viertel jeweils gleiche Dichteeigenschaften besitzen. Aufgrund dieser Eigentümlichkeit besitzt der keltische Wackelstein eine *Vorzugsrichtung*, in die er sich dreht, sobald man ihn an der Spitze nach unten drückt. Die Vorzugsrichtungen der beiden Ellipsoidhälften sind dabei genau entgegengerichtet. Dreht man den Wackelstein in Gegenwart von Reibung auf einer Ebene entgegen seiner Vorzugsrichtung, so nimmt die Winkelgeschwindigkeit immer weiter ab und der Wackelstein setzt schließlich die Rotationsbewegung in seiner Präferenzrichtung fort. Die Bezeichnung Wackelstein rührt daher, daß in der Nähe des Umkehrpunktes eine *Oszillationsbewegung* entlang der kleineren horizontalen Achse einsetzt, welche kurz vor dem Wechsel der Rotationsrichtung ihre maximale Amplitude erreicht. Mit Hilfe unserer Simulation konnten wir sowohl den Wechsel der Drehrichtung (Abbildung 6.7) als auch die Oszillationsbewegung (Abbildung 6.7) nachvollziehen.

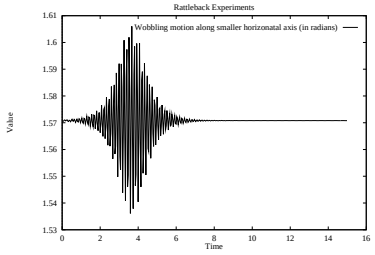
6.2.5 Vergleich von impulsbasierten und zwangsbasierten Simulationsergebnisse

Anhand des Experimentes mit dem Steh-Auf-Kreisel haben wir die Resultate der impulsbasierten Simulationsmethode mit den Daten des zwangsbasierten Verfahrens von

Abbildung 6.7 Visualisierung der Dreh- und Oszillationsbewegung anhand der Daten der Wackelsteinsimulation.



Der Wechsel der Drehrichtung.



Die Oszillationsbewegung.

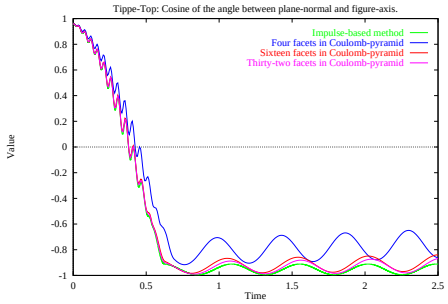
SAUER [SS98a, SS98b] verglichen. Dies war sehr einfach möglich, da beide Methoden auf dem gleichen Reibungsmodell beruhen und somit durch die gleichen Parameter beschrieben werden. Daneben erlaubte dieses spezielle Experiment, alle Stöße als plastisch anzunehmen, so daß die Systembeschreibung für beide Verfahren identisch war.

Dennoch wollen wir an dieser Stelle in Erinnerung rufen, daß den beiden Simulationsmethoden grundsätzlich verschiedene Kontaktmodelle zugrundeliegenden (vgl. Abschnitt 2.2). Um die Simulationsergebnisse besser verstehen zu können, müssen wir uns an die Integration des Reibungsmodells in das jeweilige Kollisionsauflösungsverfahren erinnern. Während im impulsbasierten Modell die Coulombschen Reibungsgesetze unmittelbar in die zu integrierenden Differentialgleichungen eingehen, berücksichtigt das zwangsbasierte Verfahren den Haftreibungsfall durch Linearisierung (Facettierung) des Reibungskegels. Haftreibung wird also umso präziser modelliert, je feiner die Facettierung dieses Kegels ist. Diese Beobachtung besitzt in dem durchgeführten Experiment große Bedeutung, da Reibungskräfte für den zu beobacht enden Effekt eine zentrale Rolle spielen.

In Abbildung 6.8 wird der Winkel zwischen der Figurenachse, um die der Kreisel dreht, und der Normalen der Auflagefläche betrachtet. Der Kreisel stellt sich also auf den Stiel, wenn der Cosinus des Winkels negativ wird. Dies geschieht etwa zum Zeitpunkt $t = 0,4$. Die Abbildung gibt dabei die ermittelten Daten der beiden Simulationsverfahren wieder. Es ist zu beobachtenden, daß sich die Resultate des zwangsbasierten Verfahrens mit zunehmender Genauigkeit der Reibungsmodellierung den Daten des impulsbasierten Verfahrens annähern und im kritischen Bereich bis $t = 0,8$ sogar übereinstimmen. Eine detailliertere Darstellung der Ergebnisse findet sich in [SSL98]

Im Rahmen der Interpretation ist selbstverständlich zu berücksichtigen, daß die hier betrachtete Problemstellung die Stärken des impulsbasierten Modells in besonderem Maße herausstellt, so daß die Annäherung der Simulationsdaten bei höherer Präzision des zwangsbasierten Verfahrens nicht als generelle Überlegenheit der impulsbasierten Simulationsmethode gewertet werden darf. Dennoch sind diese Ergebnisse mehr

Abbildung 6.8 Der Cosinus des Winkels zwischen der Figurenachse des Kreisels und der Normalen der Auflagefläche.



als erstaunlich, wenn man die grundsätzlich verschiedene Vorgehensweise der beiden Methoden in seine Überlegungen einbezieht.

7 Ausblick

Das in dieser Arbeit vorgestellte Dynamiksimulationssystem ist als Bestandteil der SILVIA-Bibliothek mit dem Ziel entwickelt worden, bestehende CAD-Entwurfssysteme um Simulations- und Evaluationswerkzeuge zu ergänzen. Dabei lag der Schwerpunkt dieser Arbeit auf der Realisierung eines in sich abgeschlossenen impulsbasierten Simulationsmoduls, was neben der Implementierung des impulsbasierten Kollisionsauflösungsverfahrens die Entwicklung eines geeigneten Kollisionserkennungssystems erforderte. Aspekte des zwingend erforderlichen Datenaustauschs zwischen CAD-System und Simulationsmodul sind dabei bisher vernachlässigt worden. Dieser Datenaustausch besitzt jedoch gerade in dem avisierten Einsatzgebiet große Bedeutung, da das Virtual Prototyping-Konzept ein weitgehendes Zusammenspiel zwischen computergestütztem Entwurf und virtueller Designerprobung erfordert.

In diesem Zusammenhang erscheint die von uns gewählte polygonale Randrepräsentation als größte Einschränkung des Konzepts. Da CAD-Systeme im allgemeinen mit gekrümmten Oberflächen wie beispielsweise NURBS arbeiten, ist die Konvertierung der Ausgangsdaten in eine Polyederapproximation unumgänglich. Dieser Konvertierungsprozeß bringt zahlreiche Probleme mit sich, die die Umsetzung und Anwendung des Virtual Prototyping-Konzepts in der Praxis erschweren.

- Die Konvertierung behindert den Datenfluß vom CAD-System zur Simulationskomponente.
- Die im Rahmen der Konvertierung erzeugte Approximation der CAD-Objekte mittels Polyeder erhöht die Beschreibungskomplexität der Objekte, was sich negativ auf die Laufzeit der Kollisionserkennung auswirkt.
- Selbst kommerzielle Konvertierungswerkzeuge erzeugen vielfach keine korrekten Polyederapproximationen, da sie nicht immer in der Lage sind, den Zusammenhang der Randpolygone sicherzustellen.
- Eine polygonale Approximation kann als Modifikation der geometrischen Struktur die zu evaluierenden Objekteigenschaften beeinflussen. So ist es beispielsweise möglich, daß die Montierbarkeit von approximierten Bauteilen eingeschränkt oder gar nicht möglich ist, während das Originalobjekt diese Eigenschaft sehr wohl besitzt.
- Die im Rahmen der Simulation gewonnenen Erkenntnisse lassen sich aufgrund der Verwendung von Approximationen nur eingeschränkt auf die ursprünglichen CAD-Objekte übertragen. So müssen beispielsweise polygonale Regionen,

die im Rahmen der Montagesimulation als kritisch identifiziert wurden, den entsprechenden Randregionen des Originalobjektes zugeordnet werden.

Diese Überlegungen verdeutlichen, daß es Sinn macht, die impulsbasierte Dynamiksimulation auf gekrümmte Oberflächen zu erweitern.

Literaturverzeichnis

[Bar89] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Comput. Graph*, 23(3):223–232, 1989. Proc. SIGGRAPH '89.

[Bar90] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Comput. Graph.*, 24(4):19–28, 1990.

[Bar92] D. Baraff. *Dynamic simulation of non-penetrating rigid bodies*. Ph.D. thesis, Cornell University, March 1992.

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

[Bar97] D. Baraff. An introduction to physically based modeling: rigid body simulation I—unconstrained rigid body dynamics. Lecture Notes, 1997.

[BBZ90] B. von Herzen, A. H. Barr, and H. R. Zatz. Geometric collisions for time-dependent curved surfaces. In *Proceedings ACM SIGGRAPH*, volume 24, pages 39–48, 1990.

[BCG+96] G. Barequet, B. Chazelle, L.J. Guibas, J.S.B. Mitchell, and A. Tal. Boxtree: A hierarchical representation for surfaces in 3d. *Eurographics*, 15(3):26–30, 1996.

[BD92] C. Bajaj and T. K. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal of Computing*, 21:339–364, 1992.

[Beh99] C. Behrens. Solid modeling mittels boolscher operationen. Master's thesis, Department of Computer Science, Saarland University, 1999.

[BK94] V. Bhatt and J. Koechling. Classifying dynamic behavior during three dimensional frictional rigid body impact. In *International Conference on Robotics and Automation*. IEEE, May 1994.

[BK96a] V. Bhatt and J. Koechling. Partitioning the parameter space according to different behavior during 3d impacts. *Transactions of ASME: Journal of Applied Mechanics*, 1996.

[BK96b] V. Bhatt and J. Koechling. Three dimensional frictional rigid body impact. *Transactions of ASME: Journal of Applied Mechanics*, 1996.

[Bra52] C. M. Braams. On the influence of friction on the motion of a top. *Physica*, 18:503–514, 1952.

[BS98a] M. Buck and E. Schömer. Interactive rigid body manipulation with obstacle contacts. *Journal of Visualization and Computer Animation*, 9:243–257, 1998.

[BS98b] M. Buck and E. Schömer. Interactive rigid body manipulation with obstacle contacts. In 6^{th} *Int. Conference in Central Europe on Computer Graphics and Visualization, WSCG'98*, pages 49–56, 1998.

[Cam55] A. R. Del Campo. Tippe top (topsy-turnee top) continued. *American Journal of Physics*, 23:544–545, 1955.

[Cam97] S. Cameron. Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *IEEE Int. Conf. Robotics and Automation*, 1997.

[Can94] J. F. Canny. Collision detection for moving polyhedra. Technical Report Memo 806, MIT A.I. Lab, 1994.

[Cha84] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithms. *SIAM Journal of Computing*, 13:488–507, 1984.

[CK86] R. K. Culley and K. G. Kempf. A collision detection algorithm based on velocity and distance bounds. In *IEEE International Conference of Robotics and Automation*, pages 1064–1068, 1986.

[CLMP95] J. Cohen, M. C. Lin, D. Manocha, and K. Ponamgi. I-Collide: An interactive and exact collision detection system for large-scaled environments. In *Proc. ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.

[CLR94] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. The MIT Press, McGraw-Hill Book Company, 14 edition, 1994.

[Coh77] R. J. Cohen. The tippe top revisited. *American Journal of Physics*, 45:12–17, 1977.

[CP89] B. Chazelle and L. Palios. Triangulating a non-convex polytope. In *Proceedings of the 5th Annual Symposium on Computational Geometry*, page 393, 1989.

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

[DE98] Johnson D. and Cohen E. A framework for efficient minimum distance computation. In *In IEEE Conference on Robotics and Automation*, pages 3678–3683, 1998.

[DHKS93] D. P. Dopkin, J. Hershberger, D. G. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[DK83] D. P. Dopkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.*, 27:241–253, 1983.

[DMY93] D. Dobrindt, K. Mehlhorn, and M. Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *Proc. of 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 314–324. Springer Verlag, 1993.

[Eck98] J. Eckstein. *Echtzeitfähige Kollisionserkennung für Virtual Reality Anwendungen*. Ph.D. thesis, Department of Computer Science, Saarland University, 1998.

[ES99] J. Eckstein and E. Schömer. Dynamic collision detection in virtual reality applications. In 7^{th} *International Conference in Central Europe on Computer Graphics and Visualization and Interactive Digital Media*, pages 71–78, 1999.

[ESCD99] Larsen E., Gottschalk S., Lin M. C., and Manocha D. Fast proximity queries with swept sphere volumes. Technical report, Department of Computer Science, University of North Carolina - Chapel Hill, 1999.

[FKN80] H. Fuchs, Z. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. In *Proc. ACM SIGGRAPH*, volume 14, pages 124–133, 1980.

[Fli96] T. Fließbach. *Mechanik*. Spektrum Akademischer Verlag Heidelberg Berlin Oxford, 1996.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice, second edition in C*. Addison-Wesley, Singapore;Tokyo;Madrid, 1996.

[Gas93] M.-P. Gascuel. An implicit formulation of precise contact modeling between flexible solids. In *Computer Graphics Proceedings, Annual Conference Series*, volume 27, pages 313–320, 1993.

[GH86] A. Garcia and M. Hubbard. Spin reversal of the rattleback: theory and experiment. In *Proceedings of the Royal Society, London*, 1986.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Computer Graphics*, pages 171–180, August 1996. Proc. SIGGRAPH'96.

[Hah88] J. K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[HDLM96] M. Hughes, C. DiMattia, M. C. Lin, and D. Manocha. Efficient and accurate interference detection for polyhedral deformation. In *Proceedings Computer Animation Conference*, 1996.

[Hen91] D. Henrich. On-line Kollisionserkennung mit hierarchisch modellierten Hindernissen für ein Mehrarm-Robotersystem. Master's thesis, Universität Karlsruhe, Fakultät für Informatik, Dezember 1991.

[HKL+99] G. Hotz, A. Kerzmann, C. Lennerz, R. Schmid, E. Schömer, and T. Warken. Silvia–a simulation library for virtual reality applications. In *procedings of IEEE Virtual Reality*, page 82, 1999.

[HKM95] M. Held, J. T. Klosowski, and J. S. B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Proc. 7$^{\text{th}}$ Canadian Conference on Computational Geometry*, pages 205–210, 1995.

[HKM+96] M. Held, J. T. Klosowski, J. S. B. Mitchell, H. Sowizral, and K. Zirkan. Efficicient collision detection using bounding volume hierarchies of k-DOPs. In *ACM SIGGRAPH'96 Visual Proceedings New Orleans*, August 1996.

[Hof89] C. M. Hoffmann. *Geometric and solid modeling*. Morgan Kaufmann Publishers, 1989.

[HS85] D. S. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10:180–184, 1985.

[Hub95] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Trans. on Visual. and Comput. Graph.*, 1(3):218–230, September 1995.

[Hug52] N. M. Hugenholtz. On tops rising by friction. *Physica*, 18:515–527, 1952.

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

[Kel86] J. B. Keller. Impact with friction. *Journal of Applied Mechanics*, 1986.

[Kle99] B. Kleineidam. Verteilte Protokolle zur effizienten Simulation von Mehrkörpersystemen. Master's thesis, Department of Computer Science, Saarland University, 1999. in preparation.

[Kon98] P. Konečný. Bounding volumes in computer graphics. Master's thesis, Masaryk University, Brno, Faculty of Informatics, April 1998.

[KPLM98] S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical shells: A higher order bounding volume for fast proximity queries. In *Proc. 1998 Workshop Algorithmic Found. Robot.*, 1998. `http://www.cs.unc.edu/~dm/collision.html`.

[KSTK98] Y. Kitanura, A. Smith, H. Takemura, and F. Kishino. A real-time algorithm for accurate collision detection for deformable polyhedral objects. *PRESENCE*, 7(1), 1998. MIT Press.

[KZ97] P. Konečný and K. Zikan. A lower bound of distance in 3d. In *Proc of WSCG*, volume 3, pages 640–649, 1997.

[LC91] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proc. IEEE Internat. Conf. Robot. Autom.*, volume 2, pages 1008–1014, 1991.

[Lem65] C. E. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689, 1965.

[Leu96] K. Chung Tat Leung. An efficient collision detection algorithm for polytopes in virtual environments. Master's thesis, University of Hong Kong, Department of Computer Science, 1996.

[Lin82] A. Lingas. The power of non-rectilinear holes. In *Proc. of 9th Colloquium on Automata , Languages and Programming*, pages 369–383. Springer-Verlag, 1982.

[Lin93] M. C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, University of California, Berkeley, CA, December 1993.

[LM93] M. C. Lin and D. Manocha. Interference detection between curved objects for computer animation. In *Models and Techniques in Computer Animation*, pages 43–57. Springer-Verlag, 1993.

[LM95] A. D. Lewis and R. M. Murray. Variational principles for constrained systems: theory and experiment. *International Journal of Nonlinear Mechanics*, 1995.

[Löt81] P. Lötstedt. Coulomb friction in two-dimensional rigid body systems. *Zeitschrift für Angewandte Mathematik und Mechanik*, 61(12):605–615, 1981.

[Löt82] P. Lötstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[Löt84] P. Lötstedt. Numerical simulation of time-dependent contact and friction problems in rigid body dynamics. *SIAM Journal of Scientific Statistical Computing*, 5(2):370–393, June 1984.

[Lum85] V. J. Lumelsky. On fast computation of distance between line segments. *Information Processing Letters*, 21:55–61, August 1985.

[MC95] B. Mirtich and J. Canny. Impulse-based dynamic simulation. In K. Goldberg, D. Halperin, J. C. Latombe, and R. Wilson, editors, *The Algorithmic Foundations of Robotics*. A. K. Peters, Wellesley, MA, 1995.

[Meg83] N. Meggido. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal of Computing*, 12:211–215, 1983.

[Mey86] W. Meyer. Distances between boxes: applications to collision detection and clipping. In *Proceedings of the 1986 IEEE International Conference on Robotics*, pages 597–602, 1986.

[Mir95] B. Mirtich. Hybrid simulation: Combining constraints and impulses. In *Workshop on Simulation and Interaction in Virtual Environments*, pages 153–158. Office of Naval Research, July 1995.

[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.

[Mir97a] B. Mirtich. Efficient algorithms for the two-phase collision detection. Technical Report TR97-23, MERL, 201 Broadway, Cambridge, MA 02139, USA, December 1997.

[Mir97b] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, MERL, 201 Broadway, Cambridge, MA 02139, USA, July 1997. `http://www.merl.com/reports/TR97-05/index.html`; submitted to ACM ToG.

[MN99] K. Mehlhorn and S. Näher. LEDA*, a platform for combinatorial and geometric computing*. Cambridge University Press, 1999.

[MS85] K. Mehlhorn and K. Simon. Intersecting two polyhedra one of which is convex. In *Proc. Found. Comput. Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 534–542. Springer Verlag, 1985.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Comput. Graph*, 22(4):289–298, August 1988.

[NAB86] I. Navazo, D. Ayala, and P. Brunet. A geometric modeller based on the exact octree representation. *Computer Graphics Forum*, 5(2):91–104, June 1986.

[NM65] J. A. Nelder and R. Mead. *Computer Journal*, 7:305–309, 1965.

[Nol96] W. Nolting. *Grundkurs Theoretische Physik*. Verlag Zimmermann-Neufang Ulmen, 1996.

[O'R85] J. O'Rourke. Finding minimal enclosing boxes. *Int. Journal of Computer and Information Sciences*, 14(3):183–199, 1985.

[Or94] A. C. Or. The dynamics of a tippe top. *SIAM Journal of Applied Mathematics*, 54(3):597–609, 1994.

[Ove92] M. H. Overmars. Point location in fat subdivisions. *Inform. Process. Lett.*, 44:261–265, 1992.

[PG95] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Proc. Eurographics*, 14(2):105–116, 1995.

[Pli54] W. A. Pliskin. The tippe top (topsy-turvy top). *American Journal of Physics*, 22:28–32, 1954.

[PML95] K. Ponamgi, D. Manocha, and M. Lin. Incremental algorithms for collision detection between solid models. In *Proc. ACM SIGGRAPH Symp. on Solid Modelling*, pages 293–304, 1995.

[PTVF92] W. H. Press, A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes In C*. Cambridge University Press, Cambridge, England, 1992.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. Int. Conf. on Robotics and Automation*, pages 3324–3329, 1994.

[Rab94] R. Rabbitz. Fast collision detection of moving convex polyhedra. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 83–109. Academic Press, Inc, 1994.

[Sau95] J. Sauer. Allgemeine Kollisionserkennung und Formrekonstruktion basierend auf Zellkomplexen. Master's thesis, Department of Computer Science, Saarland University, 1995.

[Sch94] E. Schömer. *Interaktive Montagesimulation mit Kollisionserkennung*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[Sch99] R. Schmid. Facettierung und Momenteberechnung für verallgemeinerte Polyeder. Master's thesis, Department of Computer Science, Saarland University, 1999.

[Sei90] R. Seidel. Linear programming and convex hulls made easy. In *Proc. 16th Annual ACM Symposium on Computational Geometry*, pages 211–215, 1990.

[Sha98] A. A. Shabana. *Dynamics of Multibody Systems*, chapter 6, pages 270–310. Cambridge University Press, 1998.

[SKTK95] A. Smith, Y. Kitanura, H. Takemura, and F. Kishino. A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion. In *Proceedings IEEE Virtual Annual Symposium*, pages 136–145, 1995.

[SN86] M. Szilvási-Nagy. Two algorithms for decomposing a polyhedron into convex parts. *Computer Graphics Forum*, 5(3):197–202, September 1986.

[Spa] Spatial Technology Inc., `http://www.spatial.com`. *ACIS SAT File Format*.

[SS98a] J. Sauer and E. Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, pages 153–161, 1998.

[SS98b] J. Sauer and E. Schömer. Dynamiksimulation starrer Körper für Virtual Reality Anwendungen. In *12. Symposium Simulationstechnik*, pages 355–362, 1998.

[SSL98] J. Sauer, E. Schömer, and C. Lennerz. Real-time rigid body simulations of some classical mechanics toys. In 10^{th} *European Simulation Symposium and Exhibition*, pages 93–98, 1998.

[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.

[ST96] E. Schömer and C. Thiel. Subquadratic algorithms for the general collision detection problem. In 12^{th} *European Workshop on Computational Geometry*, pages 95–101, 1996.

[Str90] W. J. Stronge. Rigid body collisions with friction. In *Proceedings of the Royal Society in London*, 1990.

[Str91] W. J. Stronge. Unraveling paradoxical theories for rigid body collisions. *Journal of Applied Mechanics*, 1991.

[SWF+93] J. M. Snyder, A. R. Woodbury, K. Fleischer, B. Currin, and A. H. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. In *Computer Graphics Proceedings, Annual Conference Series*, pages 321–334, 1993.

[Syn52] J. L. Synge. On a case of instability produced by rotation. *Philosophical Magazine*, 7:724–728, 1952.

[TPL96] J. C. Trinkle, J. S. Pang, and G. Lo. On dynamic multi-rigid-body contact problems with coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik*, 1996.

[War99] T. Warken. Berechnung von Kontaktkräften für eine zwangsbasierte Dynamiksimulation. Master's thesis, Department of Computer Science, Saarland University, 1999.

[Wel91] E. Welzl. Smallest enclosing disks (balls and ellipsoids). *Lecture Notes in Computer Science*, pages 359–370, 1991.

[WGW90] A. Witkin, Michael Gleicher, and Wilhelm Welch. Interactive dynamics. *Comput. Graph*, 24(2):11–22, March 1990.

[Zac94] G. Zachmann. Exact and fast collision detection. Master's thesis, Technische Hochschule Darmstadt, Fachbereich Informatik, Frauenhofer Institute for Computer Graphics, 1994.

[Zac97] G. Zachmann. Real-time and exact collision detection for interactive virtual prototyping. In *Proc. ASME Design Engineering Technical Conferences*, 1997.

[Zac98] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE, VRAIS'98 Atlanta*, March 1998.

[ZF95] G. Zachmann and W. Felger. The BoxTree: Enabling real time and exact collision detection of arbitrary polyhedra. In 1^{st} *Workshop on Simulation and Interaction in Virtual Environments*, pages 104–113, 1995.

[ZOMP93] J. Zyda, W. Osborne, J. Monahan, and D. Pratt. NPSNET: Real-time vehicle collisions, explosions and terrain modifications. *Journal of Visualization and Computer Animation*, 4(1):13–24, 1993.