Heuristische Bewegungsplanung mit vielen Freiheitsgraden

Thomas Chadzelek

10. März 1995

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

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

Saarbrücken, im März 1995

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Schon früh im Leben lernen Menschen, Bewegungen selbständig zu planen und zu koordinieren. Die Fähigkeit dazu wird ständig und oft unbewußt eingesetzt, nicht nur bei einer Wanderung in unbekannten Gelände. Viele der dabei gelösten Probleme sind sehr einfach, aber die mögliche Schwierigkeit von Bewegungsplanungsproblemen wird spätestens dann klar, wenn man zum ersten Mal ein Auto rückwärts einzuparken versucht oder etwa den wohlbekannten „Zauberwürfel" allzu unkontrolliert verdreht hat. Diese Aufgaben erfordern zum Teil eine sehr präzise Planung und überfordern zuweilen durch die Vielzahl der Bewegungsmöglichkeiten.

Im Bereich der Informatik traten solche Aufgaben zum ersten Mal auf, als man autonome Roboter baute, die sich in einer oft unbekannten Umgebung selbständig fortbewegen sollten, um ihre jeweiligen Aufgaben zu lösen. Damals verwandte man meist spezielle Lösungen für die konkrete Situation, ehe dann ein zunehmendes Interesse an allgemein und abstrakt formulierten Problemstellungen erwachte. Bewegungsplanung wurde zu einem Gebiet der algorithmischen Geometrie, wobei die Entwicklung heute hin zur Berücksichtigung von dynamischen Eigenschaften und weiteren Nebenbedingungen geht.

Mit zunehmender Automatisierung von Produktionsabläufen und dem Einsatz von Computern in vielen Bereichen des Lebens bieten sich zahlreiche Nutzungsmöglichkeiten für automatische Bewegungsplanungsverfahren. Dabei können in beinahe klassischer Weise Roboter in Werkshallen agieren, ohne andere Gegenstände oder gar Menschen zu gefährden; sorgfältige geplante Wege weichen selbst bewegten Hindernissen aus und können dynamisch an Veränderungen der Umgebung angepaßt werden. Ebenso aber lassen sich Montagevorgänge mit Rechnerhilfe am Bildschirm entwickeln, Einsätze von großen Maschinen auf Baustellen koordinieren oder autonome Roboter zur Erkundung lebensfeindlicher Orte einsetzen. Manche dieser Probleme sind nur einmalig zu lösen, für sie kann man Zeit und Sorgfalt aufwenden; andere Aufgaben treten in abgewandelter Weise immer wieder auf und müssen jeweils aufs Neue bearbeitet werden, vielleicht sogar unter Realzeitbedingungen.

Dennoch, und das zeigt gerade auch der Erfolg menschlichen Wegeplanens, sind viele Bewegungsplanungsprobleme intuitiv einfach und leicht zu lösen. Daher sind solche Algorithmen besonders wünschenswert, die in Abhängigkeit von der Schwierigkeit der Aufgabe schneller oder langsamer laufen und besonders auf unkomplizierte Probleme zugeschnitten sind.

Gliederung und Resultate

Diese Arbeit basiert auf einer einfachen, aber allgemeinen Strategie zur Wegesuche in Räumen beliebiger Dimension. Es handelt sich um einen heuristischen Algorithmus für das allgemeine Bewegungsplanungsproblem, der Konfigurationsräume *nicht explizit* repräsentiert oder berechnet. Die Vorgehensweise verwendet Divide-and-Conquer und läßt sich auf viele konkrete Probleme anwenden, so z.B. auf starre Körper, gelenkige Roboterarme oder die Koordination mehrerer bewegter Objekte.

Bisherige Ansätze zur Wegeplanung haben entweder zu allgemeinen, aber in der Praxis nicht handhabbaren Verfahren geführt, wie dem von Schwartz und Sharir in [SS82] beschriebenen. Oder sie haben sich darauf beschränkt, beliebige Instanzen einfacher Bewegungsplanungsprobleme effizient zu lösen, wie es Rohnert in [Ro87] für den Fall einer Kreisscheibe zwischen Polygonen in der Ebene tut. In Abkehr davon soll hier versucht werden, einfache Instanzen beliebiger Bewegungsplanungsprobleme effizient zu lösen, d.h. einen praktikablen Algorithmus zum Umgang mit praktischen Problemen zu finden. Dazu soll ein Ansatz von Schweikard aus [Sch92] untersucht und vertieft werden.

Im ersten Kapitel werden die Grundlagen des heuristischen Wegesuchverfahrens in der Ebene dargelegt und erläutert; hierbei wird auch eine Einführung in die Problemstellung und ein kurzer Überblick über bisherige Resultate gegeben. Das hier geschilderte Verfahren konstruiert einen Weg ausschließlich mittels Anfragen nach dem Schnitt einer Geraden mit dem Konfigurationsraumhindernis, wobei letzteres nicht explizit repräsentiert werden muß. Dabei werden insbesondere Beispiele schwieriger Szenen gezeigt und der Schritt von einer nichtdeterministischen zu einer deterministischen Beschreibung vollzogen. Auch die für die praktische Leistungsfähigkeit des Verfahrens wichtigen Detaillösungen hinsichtlich der Rekursionsstrategie und der Auswahl von Umwegpunkten werden dort erörtert und verfeinert; das Kapitel schließt mit einem Exkurs zur Parallelisierbarkeit der Planungsstrategie.

Der zweite Abschnitt erweitert die gewonnenen Erkenntnisse auf den Fall vieler Freiheitsgrade und schildert dessen Besonderheiten. Dort wird auch ein Verfahren entwickelt, bei dem Freiheitsgrade in Gruppen aufgeteilt werden können; dies kommt praktischen Belangen wie der effizienten Implementierung der Kollisionsrechnung sehr entgegen und bildet den Kern eines effizienten Systems zur Wegeplanung für frei im Raum bewegliche Körper. Roboter sind ein Beispiel, bei dem durch die Gelenke in natürlicher Weise eine Zusammenfassung von Freiheitsgraden jeweils eines Gelenks zu einer Einheit gegeben ist. Ebenso können mehrere abwechselnd bewegte Objekte so behandelt werden.

Anschließend zeigt das dritte Kapitel, wie der erwähnte Schnitt von Geraden mit dem Konfigurationsraumhindernis effizient berechnet werden kann, falls sich ein Polyeder im dreidimensionalen Raum translatorisch zwischen festen Hindernispolyedern bewegt. Im Gegensatz zu Schweikard verwenden wir dabei keine numerischen Approximationen, sondern eine analytische Methode ähnlich der von Canny [Ca87] erwähnten; diese wird konsequent auf leichte Implementierbarkeit und heuristische Beschleunigung durch Hüllkörper zugeschnitten. Es stellt sich heraus, daß die vollständige Information über die kollisionsfreien und -unfreien Abschnitte einer gegebenen Verschiebung in Zeit $O(n \log n)$ errechnet werden kann, wenn n das Produkt der Beschreibungskomplexitäten der beteiligten Polyeder ist, wobei nur Darstellungen der Objekte selbst benötigt werden. Dadurch wird zugleich der Vorteil dieses speziellen Ansatz unseres Planungsverfahrens bezüglich des Konfigurationsraums sichtbar wie auch eine erste konkrete Anwendung der Wegesuchstrategie.

In Ergänzung dazu wird im vierten und letzten Teil der Arbeit die Kollisionsrechnung auf Rotationen und der heuristische Bewegungsplanungsalgorithmus auf sphärische Geometrien ausgedehnt. Die Laufzeit des Verfahrens zum Schnitt von Gerade und Hindernis bleibt ebenso wie der zugrundeliegende Ansatz erhalten; es werden zur erleichterten Realisierung nur reine Rotationen betrachtet. Mit einer Trennung der translatorischen und rotatorischen Freiheitsgrade ergibt sich nun eine elegante und effiziente Strategie, um einfache Bewegungsplanungsprobleme für einen frei im Raum beweglichen Körper zu lösen. Anhand der dabei geschilderten Verfahren ist eine Umsetzung auf viele weitere Anwendungen, so auch z.B. die erwähnten Roboterarme oder auch mehrere bewegte Objekte, problemlos möglich. Das Wegesuchverfahren bildet ebenso wie die Kollisionsrechnung vielseitig einsetzbare Module, aus denen ein Algorithmus nach Bedarf erstellt werden kann.

Der Anhang schließlich umfaßt Beispiele und Statistiken in Form von Abbildungen und Tabellen; diese illustrieren das zuvor Gesagte und stellen zugleich Ergebnisse einer Implementierung der dargelegten Methoden dar.

Danksagungen

Ich bedanke mich herzlich bei Herrn Prof. Dr. Günter Hotz für die interessante Themenstellung und die großzügige Förderung sowie bei Herrn Dr. Elmar Schömer für die individuelle Betreuung. Ihre freundliche Unterstützung, ihre Geduld und ihre wertvollen Ratschläge trugen wesentlich zum Gelingen dieser Arbeit bei.

Jens Eckstein gilt mein Dank für die ausgezeichnete Zusammenarbeit im $\mathbb{I}\mathbb{G}\mathbb{O}\mathbb{R}$-Projekt und die fruchtbaren Diskussionen und Anregungen. Allen Kommilitonen und Mitarbeitern an diesem Lehrstuhl danke ich für das hervorragende Arbeitsklima.

Insbesondere aber danke ich meinen Eltern, die mir ein Studium überhaupt erst ermöglichten; ihnen und meiner ganzen Familie widme ich dieses Werk.

Kapitel 1

Eine einfache Wegesuchstrategie

1.1 Hintergrund

Bewegungsplanung wird oft der Robotik zugeschrieben, da sie die Voraussetzung für ein selbständiges Agieren eines Roboters in seiner Umgebung darstellt. Aus diesem Bedürfnis heraus hat sich das Gebiet auch entwickelt, wobei zunehmend abstrakte Aufgaben untersucht wurden, gerade auch mit Mitteln der theoretischen Informatik hinsichtlich ihrer Komplexität. In dieser Tradition orientiert sich die vorliegende Arbeit weniger an praktischen Problemen konkreter autonomer Roboter, sondern ganz generell an einem vielseitig anzuwendenden Konzept.

Das allgemeine Bewegungsplanungsproblem (engl.: *generalized movers' problem*) besteht aus der Beschreibung eines zu bewegenden Objekts inklusive Ausgangs- und Zielposition sowie eines Hindernisses. Aufgabe der Bewegungsplanung ist es nun, einen kollisionsfreien Weg für das Objekt zu finden oder zu entscheiden, daß kein solcher existiert. In dem hier betrachteten Sinn bestehen keine Einschränkungen hinsichtlich der Dynamik der Bewegung, es gilt lediglich geometrische Beschränkungen zu berücksichtigen, namentlich die Form des Objekts und des Hindernisses.

Zur Lösung dieses Problems gibt es einen ebenso allgemeinen Ansatz von Schwartz und Sharir [SS82], der auf der Methode von Collins [Col75] zur zylindrischen algebraischen Zellzerlegung beruht. Damit erreicht man eine Laufzeit polynomiell in der Beschreibungskomplexität der Szene, aber doppelt exponentiell in der Anzahl der Freiheitsgrade. Da die auftretenden Konstanten zudem sehr groß sind, hat die Vorgehensweise keine praktische Bedeutung. Canny gelang es, einen einfach exponentiellen Algorithmus zu entwickeln [Ca87], doch das ist für große Probleme immer noch inakzeptabel.

Viele erfolgreiche Versuche wurden unternommen, für spezielle Fälle effiziente Algorithmen anzugeben oder auch die inhärente Komplexität von Bewegungsplanungsproblemen zu ergründen. Effiziente Verfahren für allgemeine Probleme sind aber nicht abzusehen bzw. wegen der bereits gezeigten unteren Schranken nicht zu erwarten. Um dennoch zumindest intuitiv einfach erscheinende Probleme, selbst wenn sie sich in komplexen Szenen mit vielen Freiheitsgraden stellen, leicht und schnell zu lösen, muß man neue Wege beschreiten. Dazu soll hier ein Ansatz von Schweikard aus [Sch92] untersucht und vertieft werden.

Ein Grund für die langen Laufzeiten „herkömmlicher“ Methoden ist, daß exakte und vollständige Informationen über den Bewegungsspielraum des Objekts in seiner Umgebung errechnet werden müssen. Dabei wird typischerweise das Objekt zu einem Punkt geschrumpft, wobei sich das Hindernis gleichzeitig ausdehnt; die Schwierigkeit besteht dann darin, die genaue Form des neuen Hindernisses und des für den Punkt verbleibenden Freiraums zu errechnen. Die Repräsentation dieser Information verursacht zudem in komplexen Szenen einen hohen Speicherplatzbedarf.

In Abkehr von diesem Verfahren soll das nun betrachtete **keine explizite** Darstellung dieses neuen Hindernisses verwenden; vielmehr wird versucht mit einem Minimum an Information über die Umgebung einen Weg durch diese zu planen. Präzise gesprochen wird lediglich der Schnitt einer Geraden mit dem Konfigurationsraumhindernis betrachtet. Darin liegt der Grund für die Effizienz dieser Strategie bei höheren Dimensionen, wie später bei der Bewegung eines Körpers deutlich wird.

Diese Arbeit basiert auf einer einfachen, aber allgemeinen Strategie zur Wegesuche in Räumen beliebiger Dimension. Es handelt sich um einen heuristischen, unvollständigen Algorithmus für das allgemeine Bewegungsplanungsproblem, der es vermeidet, Konfigurationsräume explizit zu repräsentieren oder zu berechnen. Die Vorgehensweise verwendet Divide-and-Conquer und läßt sich auf viele konkrete Probleme anwenden; Beispiele dazu finden sich in den späteren Kapiteln. Die Grundlagen der Methode sollen nun zunächst in der Ebene studiert werden, dort lassen sich bereits viele der später verwendeten Konzepte begründen.

1.2 Einführung

Betrachten wir zu Anfang folgende Problemstellung: Gegeben ist eine Teilmenge $H \subset \mathbb{R}^2$ der Ebene, die als *Hindernis* aufgefaßt wird, sowie ein *Ausgangs-* und *Zielpunkt* $\mathbf{a}, \mathbf{z} \in \mathbb{R}^2$. Gesucht wird ein Weg für einen Punkt von $\mathbf{a}$ nach $\mathbf{z}$, der das Hindernis H umgeht; dieser läßt sich als eine stetige Abbildung $w : [0,1] \to \mathbb{R}^2$ auffassen, wobei $w(0) = \mathbf{a}$ und $w(1) = \mathbf{z}$. Damit der Weg auch kollisionsfrei bezüglich H ist, muß gelten

$$\forall\, 0 \le t \le 1 : \; w(t) \notin H$$

Aufgabe der Bewegungsplanung ist es nun, einen solchen Weg zu finden oder zu entscheiden, daß kein solcher Weg existiert.

Aus Gründen der Laufzeit wollen wir allerdings keinen exakten Algorithmus zur Lösung dieses Problems. Vielmehr wird auf die Entscheidungsfähigkeit verzichtet, d.h. es ist erlaubt, daß die Strategie scheitert oder nicht terminiert, obwohl ein kollisionsfreier Weg von $\mathbf{a}$ nach $\mathbf{z}$ existiert. Lediglich gefundene Wege müssen auch wirklich Lösungen sein, also insbesondere kollisionsfrei. In diesem Sinne ist die hier vorgestellte Heuristik unvollständig. Durch diese Vereinfachung der Aufgabenstellung sollte es jetzt häufig möglich sein, unbelastet von dem enormen Aufwand exakter Verfahren, leicht und schnell Lösungen zu finden.

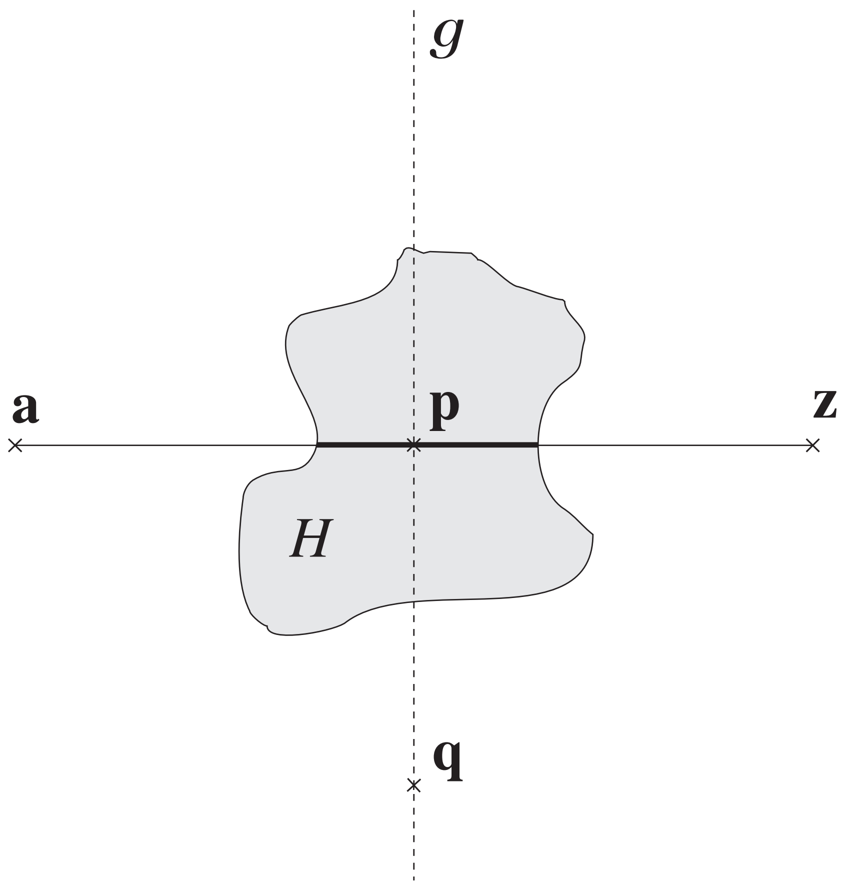


Abbildung 1.1: Ein simples Bewegungsplanungsproblem

Es erweist sich wie erwähnt als sehr günstig, jeweils nur den Schnitt von Geraden bzw. Geradensegmenten mit dem Hindernis H auszurechnen; das wird in den nächsten Kapiteln deutlich und hängt damit zusammen, daß H im allgemeinen nur implizit repräsentiert ist. Dadurch ergibt sich auch die Einschränkung, daß Wege im Sinne dieser Strategie abschnittsweise linear sind; sie lassen sich also durch ein Tupel $(\mathbf{v}_0, \ldots, \mathbf{v}_n)$ von Punkten $\mathbf{v}_i \in \mathbb{R}^2$ beschreiben. Dabei sei $n \in \mathbb{N}$ und als Weg ergibt sich

$$v : [0,1] \to \mathbb{R}^2,\ t \mapsto \begin{cases} \mathbf{v}_i & nt = i \\ \mathbf{v}_i + (nt - i)(\mathbf{v}_{i+1} - \mathbf{v}_i) & i < nt < i+1 \end{cases}$$

Daraus ersieht man unmittelbar, daß v stetig ist mit $v(0) = \mathbf{v}_0, v(1) = \mathbf{v}_n$.

Wie kann man nun solche Wege konstruieren? Betrachten wir dazu das einfache Beispiel in Abbildung 1.1. Der Versuch scheitert, $(\mathbf{a}, \mathbf{z})$ als Weg zu verwenden, da das Hindernis H die direkte Verbindung versperrt. Auf dem Liniensegment ist dabei der Schnitt mit H markiert, diese Information wird von einer Unterroutine als Antwort auf die Anfrage der Wegesuchstrategie geliefert. Es wird bis auf weiteres vorausgesetzt, daß eine solche Unterroutine existiert; später wird erklärt, wie sie effizient implementiert werden kann. Informationen über das Hindernis sind der Planungsstrategie nur durch eine derartige Anfrage zugänglich; das erlaubt viel Spielraum bei der Repräsentation von H.

Man sieht im Bild auch eine gestrichelte Gerade g, welche die Linie $(\mathbf{a}, \mathbf{z})$ im Punkt $\mathbf{p}$ schneidet. Diese Gerade liefert die entscheidende Idee zu unserem Algorithmus:

> *Jeder Weg von* $\mathbf{a}$ *nach* $\mathbf{z}$*, insbesondere jeder kollisionsfreie, muß diese Grenzlinie überschreiten, die* $\mathbf{a}$ *und* $\mathbf{z}$ *trennt. Um dem Hindernis* H *auszuweichen, kann dieser Grenzübertritt nur in einem freien Abschnitt von* g *erfolgen.*

Es ist natürlich lokal nicht entscheidbar, wo ein kollisionsfreier Weg die Grenzlinie schneiden kann, und es kommen auch kontinuierlich viele Stellen dafür in Frage. Daher wird der Schnitt von g mit H errechnet und ein Punkt auf g außerhalb von H, wie z.B. der eingezeichnete Punkt $\mathbf{q}$, wird stellvertretend für den unbekannten Ort des Grenzübertritts betrachtet. Dabei unterstellt man natürlich, daß es einen kollisionsfreien Weg gibt, und hofft weiterhin, daß auch ein solcher durch $\mathbf{q}$ existiert. Der Versuch, einen Weg zu konstruieren, wird nun rekursiv fortgesetzt, wobei zwei Teilprobleme entstanden sind: Verbinde $\mathbf{a}$ mit $\mathbf{q}$ und $\mathbf{q}$ mit $\mathbf{z}$. Dies sollte zwei Wege $(\mathbf{a}, \ldots, \mathbf{q})$ und $(\mathbf{q}, \ldots, \mathbf{z})$ liefern, die zu der Lösung $(\mathbf{a}, \ldots, \mathbf{q}, \ldots, \mathbf{z})$ zusammengesetzt werden. Hierin zeigt sich das Konzept des *Divide-and-Conquer*.

In unserem Beispiel wird schon die direkte Verbindung jeweils ein Weg von bzw. nach $\mathbf{q}$ sein, und das Problem ist mit einem Rekursionsschritt gelöst. Man sollte allerdings beachten, daß der so gefundene Weg $(\mathbf{a}, \mathbf{q}, \mathbf{z})$ zwar kollisionsfrei bezüglich H ist, aber keineswegs der kürzeste Weg für einen Punkt von $\mathbf{a}$ nach $\mathbf{z}$. Solche qualitativen Anforderungen an die gefundene Lösung, wie minimale Weglänge oder maximaler Sicherheitsabstand zum Hindernis, können generell nicht von der Heuristik berücksichtigt werden. Dies ist der Preis dafür, daß ausschließlich mit lokal begrenzter Information ein Weg gesucht wird.

Um aus dieser Idee nun ein funktionierendes Bewegungsplanungsverfahren zu machen, muß man noch zwei Punkte präzisieren: Erstens, wie bestimmt man die Gerade g, und zweitens, wo wählt man den Grenzübertritt q. Letztere Frage wird uns gleich noch beschäftigen, doch die Konstruktion der Grenzlinie läßt sich leicht beschreiben.

Als Antwort auf ihre Anfrage nach dem Schnitt der Direktverbindung mit dem Hindernis erhält die Planungsstrategie eine eventuell leere Liste von Teilstücken des Weges, die in H liegen. Auf dem ersten solchen Liniensegment, das unerlaubten Plazierungen für den zu bewegenden Punkt entspricht, wird die Mittelsenkrechte errichtet. Dadurch ergibt sich für die Planungsstrategie eine Tendenz, dem ersten Hindernis auf dem Weg auszuweichen.

Hier sind selbstverständlich weitere Möglichkeiten denkbar, z.B. auch ganz im Divide-and-Conquer-Stil ein Ausweichen auf halbem Weg von $\mathbf{a}$ nach $\mathbf{z}$. Dabei kann jeweils nur eine empirische Entscheidung getroffen werden, ob die Variante für die zu behandelnden Probleme Vorteile bringt. Einige einfache Experimente mit zufällig konstruierten Szenen aus Kreisscheiben deuten jedoch daraufhin, daß Mittelsenkrechten der obigen Art gegenüber beliebigen Senkrechten auf $(\mathbf{a}, \mathbf{z})$ vorzuziehen sind. Dies wird klarer werden, wenn weitere Details der Strategie diskutiert wurden. Sicherlich verschenkt man jedoch nichts von der Allgemeinheit des Ansatzes, wenn man sich auf Grenzgeraden senkrecht zum aktuell betrachteten Wegstück beschränkt.

Im nächsten Abschnitt wird als Einführung in die Problematik der Wahl eines Umwegpunkts q auf der Grenzgeraden g zuerst ein sehr spezieller Fall betrachtet.

1.3 Ein stets lösbarer Spezialfall

Nehmen wir einmal an, daß die Zerlegung $H = \bigcup_{i=1}^{k} H_i$ des Hindernisses in seine Zusammenhangskomponenten mit den Bedingungen

$$\begin{array}{rc} \forall\, 1 \leq i \leq k: & H_i \text{ beschränkt, offen und konvex} \\ \forall\, 1 \leq i < j \leq k: & d_{i,j} := \operatorname{dist}(H_i, H_j) > 0 \end{array} \tag{1.1}$$

gegeben ist, d.h. die Abstände zweier Komponenten seien bekannt und nicht Null.

Eine solche Zerlegung von H ist im allgemeinen nicht offensichtlich und widerspricht auch der Idee unseres Verfahrens, H nicht explizit darzustellen. Insofern ist das Szenario etwas unrealistisch, da die in dieser Arbeit eigentlich betrachtete Heuristik nicht auf solche Problemstellungen abzielt. Ferner stellen obige Forderungen starke Einschränkungen an die erlaubten Eingaben dar, die nicht wünschenswert sind.

Es läßt sich jetzt allerdings eine Wegesuchstrategie angeben (siehe [Sch92]), die nur mittels Anfragen nach dem Schnitt von Geraden mit den Hindernissen stets eine Lösung des Bewegungsplanungsproblem findet. Gegeben seien also wieder $\mathbf{a}, \mathbf{z} \in \mathbb{R}^2 \setminus H$, wobei Punkte im Hindernis natürlich eine Lösung ausschlössen. Die Vorgehensweise ist nun so, daß man sich in einer Entfernung von

$$\alpha := 1/2 \min_{1 \leq i < j \leq k} d_{i,j}$$

um Hindernisse herumtastet.

Dabei orientiert die Strategie sich an der Direktverbindung $(\mathbf{a}, \mathbf{z})$ und verläßt diesen Weg nur, um einen Bogen um eine Hinderniskomponente H_i zu machen. Danach verläuft der Pfad bis zum nächsten Hindernis wieder auf der Geraden durch Ausgangs- und Zielpunkt. Das Wegesuchverfahren entspricht hier der Strategie eines Wanderers, der eine Reihe von Seen auf seinem Weg umgeht.

Indem man also inmitten der freien Abschnitte auf $(\mathbf{a}, \mathbf{z})$ Zwischenstationen einfügt, erhält man Teilprobleme mit nur je einem zusammenhängenden Hindernis. Die Zusammensetzung der Teillösungen ergibt dieselbe Gesamtlösung, wie sie das Verfahren für das Problem mit allen Hindernissen ermittelt. Daher wird jetzt nur ein Abschnitt der Planungsphase geschildert, der ein einzelnes Hindernis H_i umgeht.

Abbildung 1.2 zeigt schematisch die Situation. Zunächst wird $(\mathbf{a}, \mathbf{z}) \cap H_i$ nachgefragt; falls kein Schnitt vorliegt, so kann der Anfangs- direkt mit dem Zielpunkt verbunden werden. Die Punkte $\mathbf{a}'$ und $\mathbf{z}'$ ergeben sich ansonsten jeweils auf der betrachteten Geraden in einem Abstand α vom Hindernis, $(\mathbf{a}, \mathbf{a}')$ und $(\mathbf{z}, \mathbf{z}')$ sind frei und werden zu einem Teil der Lösung.

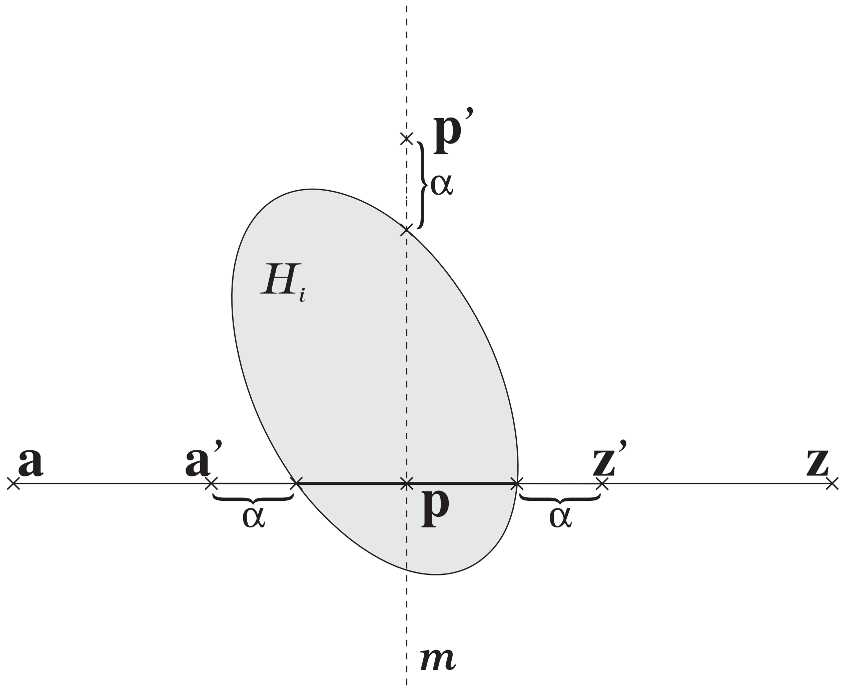


Abbildung 1.2: Ein Szene mit konvexem Hindernis

Auf dem Kollisionsabschnitt (fett gezeichnet) wird die Mittelsenkrechte m errichtet und liefert „links des Weges" den Punkt $\mathbf{p}'$, wiederum gerade α außerhalb von H_i auf m. Wie schon eben wird jetzt auch rekursiv $\mathbf{a}'$ mit $\mathbf{p}'$ und $\mathbf{p}'$ mit $\mathbf{z}'$ verbunden, wodurch sich als Weg insgesamt $(\mathbf{a}, \mathbf{a}', \ldots, \mathbf{p}', \ldots, \mathbf{z}', \mathbf{z})$ ergibt.

Zur Korrektheit des Verfahrens bleibt noch zu zeigen, daß man sich in einem Teilabschnitt wirklich auf ein einzelnes Hindernis beschränken kann.

Die Behauptung folgt daraus, daß alle Punkte $\mathbf{x}'$ und die Liniensegmente zwischen ihnen innerhalb eines „α-Abstands" vom betrachteten Hindernis H_i bleiben. Da je zwei Hindernisse aber mindestens 2α voneinander entfernt sind, können alle anderen $H_j,\ i \neq j$, vernachlässigt werden.

Was bedeutet das konkret? Für den Punkt $\mathbf{a}'$ gilt nach Konstruktion $\operatorname{dist}(\mathbf{a}', H_i) \leq \alpha$, analog für $\mathbf{z}'$ und $\mathbf{p}'$. Zu den Punkten $\mathbf{x}'$ korrespondieren jeweils $\hat{\mathbf{x}}$ auf dem Rand des Hindernisses, die auf einer Geraden mit $\mathbf{x}$ und $\mathbf{x}'$ liegen; Abbildung 1.3 verdeutlicht dies.

Die Wegesuchstrategie stellt eine Anfrage nach dem Schnitt von $(\mathbf{a}', \mathbf{p}')$ mit den Hindernissen. Dieses Liniensegment ist in der α-Umgebung des Segments $(\hat{\mathbf{a}}, \hat{\mathbf{p}})$ enthalten, denn diese Umgebung ist konvex und umfaßt die beiden Endpunkte $\mathbf{a}'$ und $\mathbf{p}'$. Es gilt jetzt aber $(\hat{\mathbf{a}}, \hat{\mathbf{p}}) \subset H_i$ wegen der Konvexität von H_i und damit

$$\begin{array}{rrl} & \forall\, \mathbf{v} \in (\mathbf{a}', \mathbf{p}')\ \exists\, \mathbf{w} \in (\hat{\mathbf{a}}, \hat{\mathbf{p}}): & \operatorname{dist}(\mathbf{v}, \mathbf{w}) \leq \alpha \\ \Rightarrow & \forall\, \mathbf{v} \in (\mathbf{a}', \mathbf{p}'): & \operatorname{dist}(\mathbf{v}, H_i) \leq \alpha \end{array}$$

so daß keiner dieser Punkte in einem anderen Hindernis als H_i liegen kann. Analoges

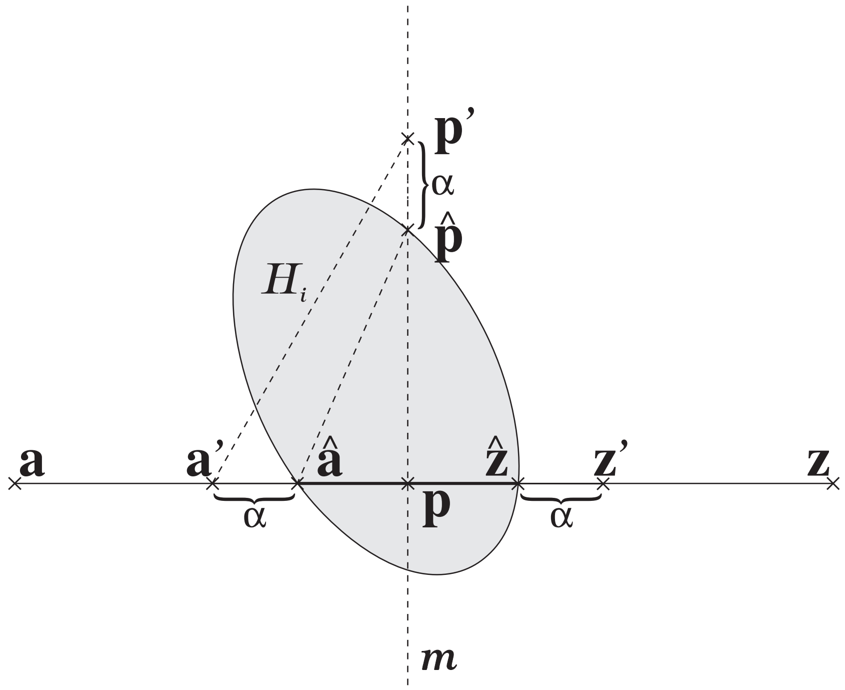


Abbildung 1.3: Linie bleibt in α-Abstand zu H_i

gilt für die Linie $(\mathbf{p}', \mathbf{z}')$.

Zuletzt soll noch angedeutet werden, warum die genannten Bedingungen 1.1 wichtig für das vorgestellte Verfahren sind. Unbeschränkte Hindernisse erlauben es nicht, einfach „links vorbei“ zu gehen, wie es eben beschrieben wurde. In diesem Falle müßte man jeweils zu einer erreichbaren Seite ausweichen und sich bei der Rekursion auf einen Quadranten der Ebene beschränken, damit alle Umwegpunkte auf derselben Seite gewählt werden.

Offenheit ist nicht notwendig für die Konstruktion der Mittelsenkrechten, vermeidet aber die Entartung eines punktförmigen Kollisionsintervalls. In der hier vorgestellten Version kommt der Algorithmus sicher auch mit abgeschlossenen Hindernissen zurecht; die Bedingung wurde von Schweikard übernommen.

Wirklich wesentlich für ein korrektes Arbeiten der Strategie sind aber konvexe Hindernisse. An einem Beispiel soll daher noch erläutert werden, wie der Algorithmus andernfalls in eine Endlosschleife geraten und damit scheitern könnte. Dadurch soll auch demonstriert werden, welche Probleme eine solche heuristische Suchstrategie mit allgemeinen Problemstellungen hat.

1.3.1 Ein „unlösbares“ Problem

Für die Konstruktion einer „unlösbaren“ Szene ist es unabdingbar, die darauf anzuwendende Strategie festzulegen. Schon kleine Änderungen am Bewegungsplanungsalgorithmus können das Problem wieder lösbar machen, d.h. auch die Aufgabenstellung müßte geändert werden. Das folgende Beispiel bezieht sich daher ausdrücklich auf die eben vorgestellte Vorgehensweise, die Mittelsenkrechte auf Hindernissen zu errichten und nach links auszuweichen, wobei ein α-Abstand eingehalten wird.

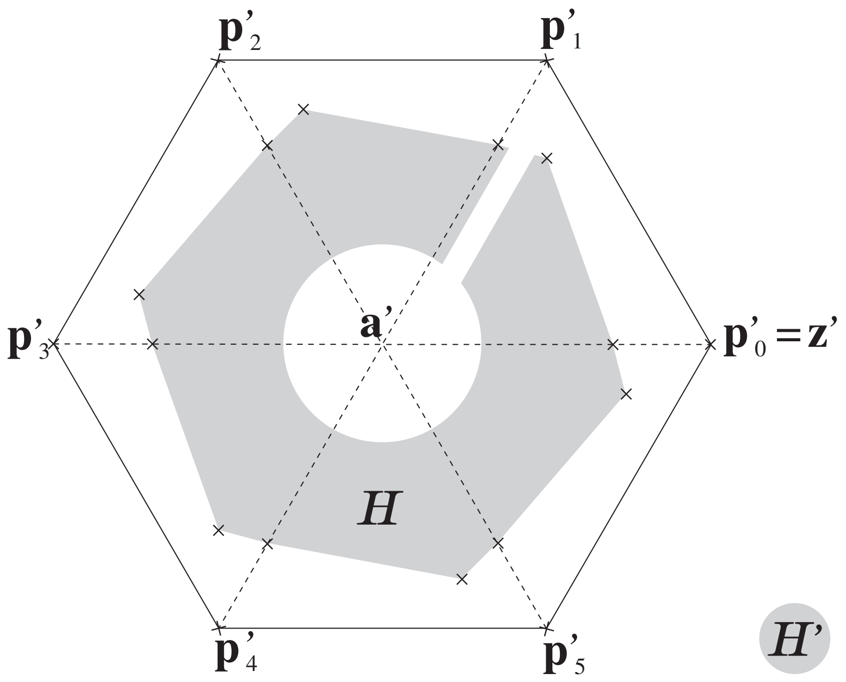


Abbildung 1.4: ENDLOSSCHLEIFE BEI NICHTKONVEXEM HINDERNIS

Die grundlegende Idee ist es, daß die Punkte $\mathbf{p}'_i$ Ecken eines regelmäßigen Sechsecks mit Mittelpunkt $\mathbf{a}'$ sind, $\mathbf{z}' = \mathbf{p}'_0$. Eine Endlosschleife entsteht dadurch, daß der Schritt vom Problem $\mathbf{a}' \to \mathbf{p}'_i$ zu $\mathbf{a}' \to \mathbf{p}'_{i+1 \bmod 6}$ die eigentliche Aufgabe nur um 60 Grad dreht und einer Lösung nicht näher kommt. Das Hindernis ist im wesentlichen ein Ring, der innen und außen Abstand α zu den Punkten $\mathbf{x}'$ hält; er wird allerdings an einer Stelle aufgeschnitten, damit ein Weg vom Start zum Ziel existiert.

Der innere Rand des Rings ist ein Kreis um $\mathbf{a}$ mit Radius α, der äußere dagegen ein unregelmäßiges Zwölfeck. Dessen Ecken sind gerade seine Schnittpunkte mit den Liniensegmenten $(\mathbf{a}', \mathbf{p}'_i)$ bzw. die Punkte $\hat{\mathbf{p}}_i$, die während des Verfahrens generiert werden. Dargestellt ist ferner eine Hinderniskomponente H', welche nur dazu dient, den Abstand $\alpha := 1/2\,\mathrm{dist}(H, H')$ zu definieren; für das eigentliche Planungsproblem spielt sie keine Rolle und wird daher nicht weiter erwähnt.

Zuerst sollte man verstehen, wie ein Schritt der Wegesuchstrategie in diesem Fall aussieht. Abbildung 1.5 zeigt wie bekannt Anfangs- und Endpunkt $\mathbf{a} = \mathbf{a}'$ und $\mathbf{z} = \mathbf{z}'$ des gesuchten Weges, das Kollisionsintervall mit dem Hindernis und die Mittelsenkrechte m darauf. Letztere schneidet den Kreis um $\mathbf{a}'$ durch $\mathbf{z}'$ „links des Weges" im Punkt $\mathbf{p}'_1$; die Ausmaße des Hindernisrings sind so gewählt, daß $\mathbf{p}'_1$ — gemessen auf m — gerade α außerhalb des Hindernisses liegt. Daher werden die beiden Teilprobleme $\mathbf{a}' \to \mathbf{p}'_1$ und $\mathbf{p}'_1 \to \mathbf{z}'$ betrachtet, wobei das zweite trivial ist, das erste aber ungelöst

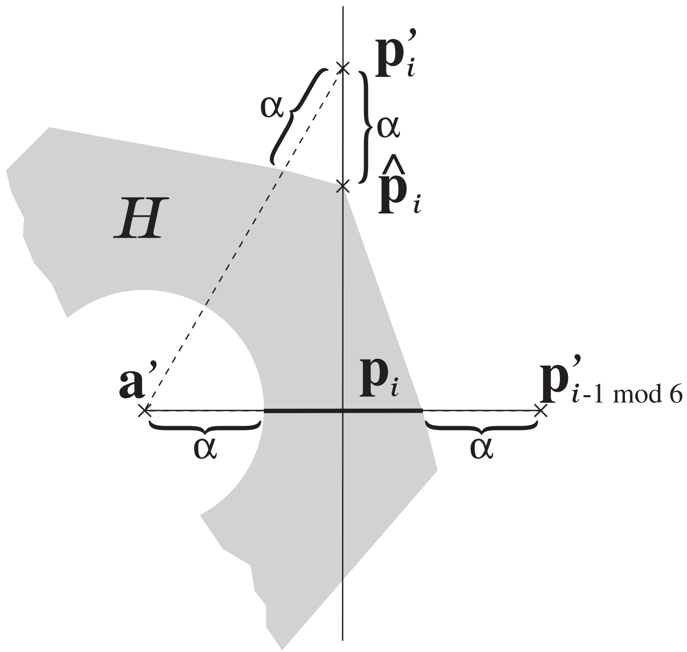


Abbildung 1.5: Ein Schritt unter unendlich vielen

bleibt. Die heuristische Vorgehensweise wird endlos versuchen, $\mathbf{a}'$ mit den Punkten $\mathbf{p}_i'$ zu verbinden, wobei i immer wieder die Werte $0, \ldots, 5$ durchläuft.

Nun macht ein vollständiger Ring die Aufgabe natürlich für jedes Planungsverfahren unlösbar, daher sehen wir eine kleine Öffnung darin vor. Wo findet sich ein Platz dafür? Die Verbindung von $\mathbf{a}'$ und $\mathbf{p}_i'$ sollte das Hindernis so schneiden, daß sich die richtige Mittelsenkrechte ergibt, zudem muß der α-Abstand nach innen und außen eingehalten werden. Die Mittelsenkrechte muß von $\mathbf{p}$ bis $\hat{\mathbf{p}}$ im Hindernis verlaufen; hier verändern wir den Ring daher auch nicht.

Betrachten wir aber den Schnittpunkt $\mathbf{s}$ von $(\mathbf{a}', \mathbf{p}_1')$ mit dem äußeren Rand des Zwölfeckrings, so erkennen wir eine geeignete Stelle. Wegen $\alpha > 0$ gilt auch $\epsilon := \mathrm{dist}(\mathbf{s}, \hat{\mathbf{p}_1}) > 0$, daher kann ein Bereich mit Radius $\epsilon/3$ um eine Parallele zu $(\mathbf{a}', \mathbf{p}_1')$ durch den Mittelpunkt von $(\mathbf{s}, \hat{\mathbf{p}_1})$ aus dem Hindernis entfernt werden, so wie es in Abbildung 1.6 angedeutet ist.

Somit ist ein lösbares Bewegungsplanungsproblem konstruiert, das von der eben dargelegten Strategie deshalb nicht gelöst werden kann, weil es ein nichtkonvexes Hindernis enthält.

1.4 Weiterentwicklung des Verfahrens

Obwohl soeben beispielhaft gezeigt wurde, daß im Falle nichtkonvexer Hindernisse nicht garantiert werden kann, daß die Suchstrategie einen existierenden Weg findet, muß man sie im allgemeinen dennoch auf solche Szenen anwenden. Bei typischen Bewegungspla-

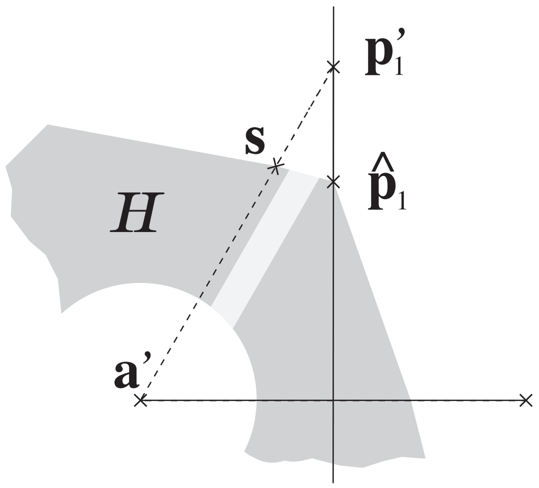


Abbildung 1.6: Aufschneiden des Kreisrings

nungsproblemen sind eben die Hindernisse nicht immer konvex; man hofft dann jedoch, daß sie auch nicht so pathologisch wie im obigen Beispiel sind. Ganz allgemein wird es jetzt darum gehen, die starken Einschränkungen 1.1 zu lockern und eine Variante des Verfahrens zu entwickeln, das dann immer noch eingesetzt werden kann.

Die Forderung nach konvexen Hindernissen wird einfach fallen gelassen, wodurch nunmehr nicht notwendig eine Lösung gefunden wird. Statistische Auswertungen einer konkreten Anwendung der Strategie zeigen jedoch, daß in halbwegs einfachen Situationen schnell und zuverlässig Wege gefunden werden. Daher wird auch nicht versucht, den Algorithmus speziell für nichtkonvexe Hindernisse zu adaptieren.

Vielmehr ist es der verwendete Abstand der Zusammenhangskomponenten des Hindernisses voneinander, der einer praktischen Anwendung bislang im Wege steht. Wie bereits erwähnt, ist dieser Wert nicht bekannt und nur aufwendig zu berechnen, zumal das Hindernis nicht explizit realisiert werden soll. Daher muß ein Ersatz für α gefunden werden.

Bisher wurden Umwegpunkte aufgrund des Wissen über Abstände nahe genug an einer Hinderniskomponente gewählt, so daß keine anderen Teile dem zu konstruierenden Weg in die Quere kommen konnten. Andererseits blieben sie auch soweit wie möglich davon entfernt, damit die Hinderniskomponente selbst effizient umrundet wird. Um in Unkenntnis von α eine ähnliche Strategie zu verfolgen, wird nun als Umwegpunkt jeweils der Mittelpunkt eines freien Abschnitts auf der Mittelsenkrechten m gewählt. Dadurch erreicht man zumindest, daß der Umwegpunkt von den bekannten Teilen des Hindernisses lokal maximalen Abstand hält.

Dabei wird vorausgesetzt, daß alle diese Freiräume zumindest durch eine äußere Grenze des Spielraums für den bewegten Punkt beschränkt sind. Dies kann explizit durch ein unbeschränktes Hindernis oder implizit dadurch geschehen, daß die erlaubten Koordinaten eingegrenzt werden.

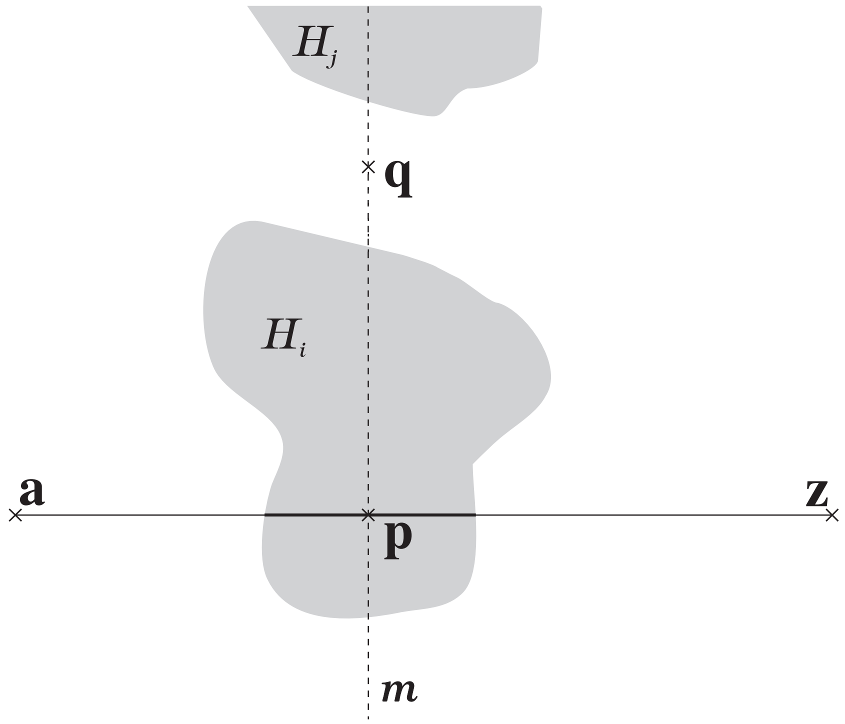


Abbildung 1.7: STRATEGIE OHNE KENNTNIS VON HINDERNISABSTÄNDEN

In Bild 1.7 erkennt man, wie der Umwegpunkt $\mathbf{q}$ konstruiert wird. Anfangs- und Zielpunkt werden unmittelbar durch eine Linie verbunden, deren Schnitt mit dem ersten Hindernis ein Kollisionsintervall liefert. Durch dessen Mitte errichtet man die Senkrechte m zu $(\mathbf{a}, \mathbf{z})$, welche „links des Weges" ein erstes Intervall außerhalb der Hindernisses enthält.

Informationen über solche Intervalle erhält die Planungsstrategie, wie bereits erwähnt, durch Anfragen an eine Unterroutine nach dem Schnitt eines Liniensegments bzw. einer Gerade mit allen Hindernissen. Dabei werden die einzelnen Zusammenhangskomponenten des Gesamthindernisses hier konzeptuell wie einzelne Hindernisse behandelt; es sei betont, daß eine solche Zerlegung bei einer Implementierung nicht errechnet werden muß. Vielmehr erlaubt es diese Sprechweise lediglich, die hier verwendeten Bilder anschaulich zu kommentieren.

Der Aufteilungsschritt des Algorithmus erzeugt nun wieder zwei Teilprobleme und konstruiert so eine Verbindung $(\mathbf{a}, \ldots, \mathbf{q})$ und eine andere $(\mathbf{q}, \ldots, \mathbf{z})$, die zusammen das Gesamtproblem lösen. Im Unterschied zu unserem ersten Ansatz wird hier also nur dem jeweils ersten Hindernis unmittelbar ausgewichen und die Lage des Umwegpunktes ist etwas anders. Diese kleinen Änderungen genügen aber, um die Planungsstrategie auf unbekannte Hindernisse anzuwenden. Ihre eigentliche Stärke zeigt sich auch genau darin, daß man eben keine aufwendigen Vorberechnungen auf den Hindernissen ausführen muß, ehe man einfache Wege planen kann.

Zum Abschluß dieses Abschnitts sei noch angemerkt, daß ein unlösbares Beispiel

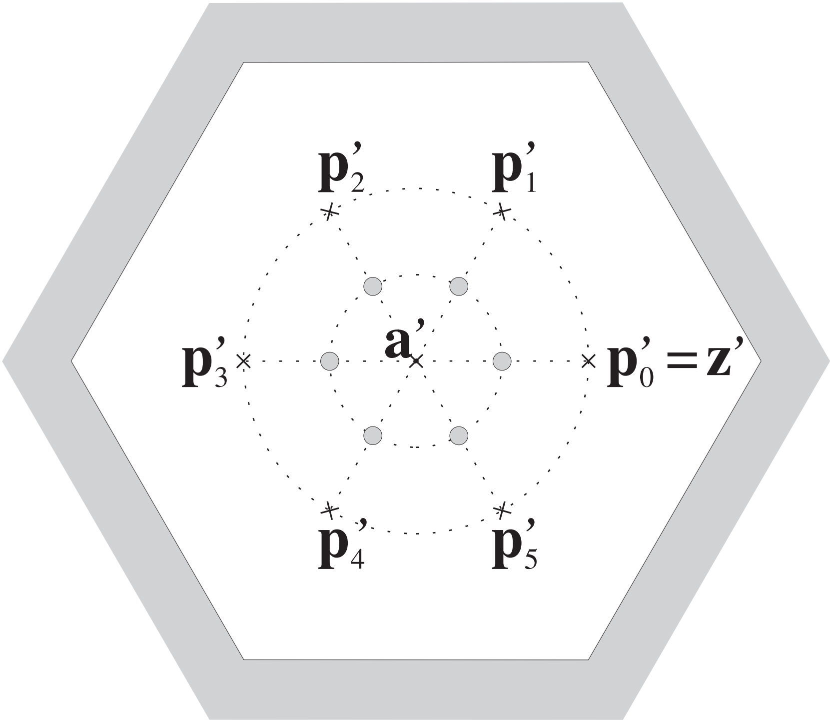


Abbildung 1.8: Variante des „unlösbaren" Problems

für diese Strategie ganz analog zu obigem konstruiert werden kann; siehe dazu auch Abbildung 1.8. Dabei wird wieder ein endlose, da zyklische Folge von Umwegpunkten erzeugt, und das Verfahren scheitert durch Nichtterminierung. Die Ausweichpositionen bilden ebenfalls ein Sechseck mit dem Startpunkt in der Mitte, als Hindernisse genügen geschickt angeordnete Kreisscheiben. Man bemerkt also, daß dieses Problem mit dem vorherigen Verfahren gelöst werden kann; durch die Streichung des Abstandswertes α verliert die Strategie also in der Tat an Stärke.

Ferner läßt sich hier auch zeigen, daß die Möglichkeit, nach rechts auszuweichen, nicht notwendig eine Verbesserung mit sich bringt. Wenn die Szene nach außen durch einen Rand begrenzt ist, ergeben sich nur zwei Intervalle auf der Geraden m, in denen man den Punkt $\mathbf{q}$ wählen kann. Diese Wahlfreiheit bestimmt in diesem Beispiel aber lediglich, ob das Sechseck im oder gegen den Uhrzeigersinn durchlaufen wird.

1.5 Nichtdeterministische Beschreibung

Wir wollen das eben erwähnte Beispiel noch einmal aufgreifen und erläutern, wie man die Wegesuchstrategie für eine nichtdeterministische Maschine beschreiben kann. In Abbildung 1.8 sieht man die Variante der Sechseckszene mit äußerem Rand und einigen Kreisscheiben. Diese sind so geschickt angeordnet, daß wiederum eine Endlosschleife entsteht, wenn Umwegpunkte in der Mitte von Freiintervallen gewählt werden. Durch eine nur leicht veränderte Wahl dieser Ausweichbewegungen findet die Strategie dagegen rasch einen Weg vom Start- zum Zielpunkt. Natürlich ließe sich dann die Szene entsprechend modifizieren, damit der Effekt erhalten bleibt. Durch mehr oder minder kompliziert konstruierte Beispiele ist sicher jede Variante dieses hier vorgestellten Schemas in die Irre zu führen, es sei denn, sie verhielte sich unvorhersehbar.

Das führt daher unmittelbar zur Idee, bestimmte Entscheidungen im Laufe des Konstruktionsverfahrens dem Zufall zu überlassen. Bei Ausführung des Planungsverfahrens auf einer nichtdeterministischen Maschine gibt es dann erfolgreiche Berechnungen, solange es überhaupt solche stückweise linearen Wege gibt. Voraussetzung dafür ist allerdings, daß die implementierte Strategie die Menge möglicher Wege nicht zu weit einschränkt.

Dabei ist es völlig unkritisch, wie die Grenzgerade g gewählt wird, solange jeder freie Punkt darauf als möglicher Ort q des Grenzübertritts zugelassen bleibt. Daher wird die Grenzgerade stets als Mittelsenkrechte auf dem ersten Kollisionsintervall errichtet.

Als allgemeine Problemstellung betrachten wir ein beliebiges Hindernis sowie zwei Punkte außerhalb davon, die es durch einen Weg zu verbinden gilt. Zur Beschreibung für letzteren dient dabei eine Liste von Punkten, welche den stückweise linearen Weg in bereits erwähnter Weise definieren. Die Strategie erzeugt die Eckpunkte des Pfades der Reihe nach und verwaltet sie durch eine Liste, welche bei jedem Aufruf mitgeführt wird; daher wird der Startpunkt schon zu Anfang übergeben.

geg.:

$H \subset \mathbb{R}^2$	Hindernis
$\mathbf{a} \in \mathbb{R}^2 \setminus H$	Anfang
$\mathbf{z} \in \mathbb{R}^2 \setminus H$	Ziel

ges.:

$pfad : \text{List}(\mathbb{R}^2)$	berechneter Weg

Aufruf:

$pfad \leftarrow \text{connect}(\mathbf{a}, \mathbf{z}, [\mathbf{a}])$	$[\mathbf{a}]$ bezeichne einelementige Liste

Damit ergibt sich folgender Algorithmus 1.9, der seine Informationen über das Hindernis nur durch Anfragen an ein Unterprogramm erhält. Zu diesem Zweck sollen zwei Funktionen, „intersectSegment“ und „intersectLine“, zur Verfügung stehen, welche den Schnitt eines Liniensegments respektive einer Geraden mit dem Hindernis H berechnen. Als Ergebnis wird eine Liste von Intervallen erwartet, wobei

$$\text{intersectSegment}(\mathbf{a}, \mathbf{z}) = \{t \in [0, 1] \mid \mathbf{a} + t\,(\mathbf{z} - \mathbf{a}) \in H\}$$

und analog

$$\text{intersectLine}(\mathbf{p}, \mathbf{r}) = \{t \in \mathbb{R} \mid \mathbf{p} + t\,\mathbf{r} \in H\}$$

Der Ablauf des Programms sieht dann so aus, daß zuerst die Direktverbindung von Ausgangs- und Zielpunkt auf Kollisionsfreiheit geprüft wird. Ist der Test erfolgreich, so wird der bereits konstruierte Teilpfad um den Zielpunkt erweitert und zurück gegeben.

Andernfalls werden Aufpunkt und Richtungsvektor der Mittelsenkrechten auf dem ersten Hindernis bestimmt, mit diesen Werten erfolgt eine Anfrage an „intersectLine“. Als Ort des Grenzübertritts wird ein zufälliger Punkt ermittelt, der nicht im Hindernis liegt; mittels dieses Umwegpunktes kann das Problem aufgeteilt werden.

connect($\mathbf{a}, \mathbf{z} : \mathbb{R}^2$; $pfad$: List($\mathbb{R}^2$)) : List($\mathbb{R}^2$)
    $\mathbf{p}, \mathbf{q}, \mathbf{r} : \mathbb{R}^2$
    L : IntervalList
    $t : \mathbb{R}$

$L \leftarrow$ intersectSegment($\mathbf{a}, \mathbf{z}$)	Kollisionsintervalle
if empty(L) **then**	
$pfad \leftarrow pfad \circ [\mathbf{z}]$	Direktverbindung ist frei
else	
$t \leftarrow$ mid(head(L))	Mittelpunkt des 1. Intervalls
$\mathbf{p} \leftarrow (1-t)\,\mathbf{a} + t\,\mathbf{z}$	mitten im 1. Hindernis
$\mathbf{r} \leftarrow (\mathbf{z} - \mathbf{a})^{\perp}$	senkrecht zur Wegrichtung
$L \leftarrow$ intersectLine($\mathbf{p}, \mathbf{r}$)	Kollisionsintervalle
randomly choose $t \notin L$	zufälliger freier Punkt
$\mathbf{q} \leftarrow \mathbf{p} + t\,\mathbf{r}$	Ausweichpunkt auf Mittelsenkrechte
$pfad \leftarrow$ connect($\mathbf{a}, \mathbf{q}, pfad$)	rekursive Lösung …
$pfad \leftarrow$ connect($\mathbf{q}, \mathbf{z}, pfad$)	… der Teilprobleme

return $pfad$

Abbildung 1.9: NICHTDETERMINISTISCHE WEGESUCHSTRATEGIE

Dieser Algorithmus ist zur Ausführung auf einer nichtdeterministischen Maschine gedacht in dem Sinne, daß eine erfolgreiche Berechnung gesucht ist. Führt man ihn als randomisierten Algorithmus auf einer realen Maschine durch, so wird er nicht immer terminieren. Dies trifft besonders dann zu, wenn der Freiraum der Szene in mehrere Zusammenhangskomponenten zerfällt und ein Umwegpunkt $\mathbf{q}$ in einer anderen Komponente als Start und Ziel gewählt wird. In diesem Fall mag es zwar einen Weg geben, aber sicherlich kann er nicht über den Punkt $\mathbf{q}$ verlaufen.

Damit hängt auch zusammen, daß die maximale Rekursionstiefe des Schemas nicht festzulegen ist; jede Grenze, die man hier ziehen könnte, wäre künstlich und verhinderte womöglich eine Lösung des Problems. Dennoch kommt man nicht umhin, die Strategie an einer endlosen Rekursion zu hindern, um im Falle des Scheiterns eine neue Berechnung auszuprobieren. Im Folgenden soll nun erläutert werden, wie ein deterministisches Verfahren versuchen kann, erfolgreiche Berechnungen der nichtdeterministischen Maschine aufzuspüren.

1.6 Deterministische Beschreibung durch Backtracking

Die Frage ist also, wie man die Menge der möglichen Berechnungen der nichtdeterministischen Maschine effizient durchsucht, um eine erfolgreiche Rechnung zu finden. Dabei stellt sich zuerst einmal das Problem, daß das zuvor geschilderte Verfahren schon bei der Wahl eines Umwegpunktes kontinuierlich viele Möglichkeiten bietet. Es ist daher unmöglich, alle Elemente des Suchraums aufzuzählen, und in einem realen Computer mit endlichem Speicher kann nicht einmal jede einzelne Rechnung exakt dargestellt werden.

Insgesamt ist die hier untersuchte Planungsstrategie auch kein stetiger Vorgang, d.h. die gefundenen Lösungen können sich schon bei kleinen Variationen von Ausgangs- und Zielpunkt sprunghaft ändern. Dies geschieht zum Beispiel, wenn die Direktverbindung plötzlich ein neues Hindernis tangiert und die Grenzgerade dadurch an einer ganz anderen Stelle betrachtet wird. Daher ist auch nicht zu erkennen, wie man eine Partitionierung der Umwegpunkte in endlich viele Äquivalenzklassen erreichen könnte, um wenigstens auf diese Weise keinen Punkt völlig außer Acht zu lassen.

Wie kann man nun zu einer Approximation des Suchraums gelangen, die möglichst viele erfolgreiche Rechnungen beinhaltet und dennoch leicht durchsucht werden kann? Unter diesem Blickwinkel ist es eine gute Idee, bei der Anpassung der Strategie, wie im Abschnitt 1.4 beschrieben, als Umwegpunkt jeweils nur den Mittelpunkt eines freien Intervalls auf der Grenzgeraden zuzulassen. Diesem Ansatz liegt zuerst einmal die fragwürdige Annahme zugrunde, daß innerhalb eines freien Liniensegments außerhalb des Hindernisses alle Punkte gleich gut erreichbar sind; das sollte allerdings auch in etwa gewährleistet sein, da unser Planungsverfahren sie unmittelbar miteinander verbinden kann. Dadurch ergibt sich eine Anzahl von möglichen Ausweichpositionen auf einer Grenzgeraden, die durch die Beschreibungskomplexität der Hindernisse beschränkt wird. Unter geeigneten Voraussetzungen sind das bereits nur endlich viele, aber es empfiehlt sich dennoch im allgemeinen, die Anzahl weiter einzuschränken. Zu diesem Zweck kann eine Bewertungsfunktion verwendet werden, mit deren Hilfe jeweils die besten k Umwegpunkte ausgewählt werden, wobei k eine vorgegebene Konstante ist.

Um nun mögliche Berechnungen der nichtdeterministischen Maschine aufzuzählen, kann man die möglichen Umwegpunkte auf einer Grenzgerade errechnen und systematisch durchprobieren. D.h. die rekursiven Aufrufe werden für den ersten Punkt ausgeführt und signalisieren, ob das Verfahren scheiterte; in diesem Fall wird dann der nächste Punkt erprobt. Im Erfolgsfall, oder falls keiner der Kandidaten zu einer Lösung führte, erfolgt eine entsprechende Rückmeldung. Dieses Verfahren von Versuch und Irrtum ist auch unter dem Namen *Backtracking* bekannt, denn im Falle des Scheiterns werden Entscheidungen revidiert und der Weg in die Sackgasse zurückverfolgt.

Woran kann die Strategie zur Wegesuche nun konkret scheitern? Dies können zum einen Endlosschleifen sein, wie sie in den bisherigen Beispielen konstruiert wurden. Zum anderen aber, und das tritt in der Praxis weitaus häufiger auf, kann ein Ausweichpunkt $\mathbf{q}$ gewählt werden, von dem aus es gar keinen Weg zu Start und Ziel gibt. Dies trifft wie erwähnt dann zu, wenn der Freiraum der Szene in mehrere Zusammenhangskomponenten zerfällt und $\mathbf{q}$ in einer anderen Komponente als Start und Ziel gewählt wird.

Diese Situation kann allerdings nicht innerhalb des Programms erkannt werden, da keine globalen Informationen über das Hindernis verwendet werden. So bietet sich nur die Möglichkeit, die erlaubte Rekursionstiefe zu beschränken und ein Überschreiten als Scheitern anzusehen. Damit werden sämtliche Endlosschleifen vermieden, aber selbstverständlich auch einige erfolgreiche Berechnungen übersehen. Es verbleibt in diesem Zusammenhang noch die Frage, wie die zulässige Rekursionstiefe gewählt werden soll.

Abbildung 1.10 zeigt den modifizierten Bewegungsplanungsalgorithmus mit Beschränkung der Rekursionstiefe; analog zur nichtdeterministischen Beschreibung ergeben sich Aufgabenstellung und Aufruf wie folgt.

geg.:

$H \subset \mathbb{R}^2$	Hindernis
$\mathbf{a} \in \mathbb{R}^2 \setminus H$	Anfang
$\mathbf{z} \in \mathbb{R}^2 \setminus H$	Ziel
$r_{max} \in \mathbb{N}$	maximale Rekursionstiefe

ges.:

$pfad : \mathrm{List}(\mathbb{R}^2)$	berechneter Weg

Aufruf:

$pfad \leftarrow [\mathbf{a}]$	einelementige Liste
$\mathrm{connect}(\mathbf{a}, \mathbf{z}, r_{max}, pfad)$	liefert **true** gdw. erfolgreich

Diese Prozedur verwendet eine neue Unterroutine namens „viaPoints“, welche eine Liste der auszuprobierenden Umwegpunkte zurückgibt. Zu diesem Zweck steht die Information über Anfangs- und Zielpunkt sowie Kollisionsintervalle zur Verfügung. Bei der Implementierung dieser Funktion ergeben sich daher einige Freiheiten, welche Grenzgerade betrachtet und wo Umwegpunkte auf ihr festgelegt werden; ferner kann deren Anzahl und Anordnung gewählt werden. Diesem Themenkomplex widmet sich der übernächste Abschnitt, nachdem auf die Bestimmung der Rekursionstiefe eingegangen wurde.

1.7 Wahl der Rekursionstiefe

Das eben geschilderte Backtrackingverfahren reduziert das Suchproblem in einem Raum von überabzählbarer Kardinalität auf das Durchsuchen endlich vieler möglicher Berechnungen. Eine solche Berechnung des Divide-and-Conquer-Algorithmus läßt sich als binärer Baum darstellen, wobei die Knoten jeweils einem Bewegungsplanungsproblem und damit einer Instanz der Funktion „connect“ entsprechen, die Kanten hingegen den rekursiven Aufrufen zur Zerteilung des Problems. In diesem Modell entsprechen Blätter gerade jenen Aufgabenstellungen, welche durch die Direktverbindung von Ausgangs- und Zielpunkt gelöst werden. Um Endlosschleifen zu vermeiden, wurde eine Begrenzung für die Rekursionstiefe der Strategie eingeführt; d.h. die Suche nach

```
connect(a, z : ℝ²; rek : ℕ; var pfad : List(ℝ²)) : Boolean
    q : ℝ²
    L : IntervalList
    pfad′, via : List(ℝ²)
    ok : Boolean

L ← intersectSegment(a, z)                              Kollisionsintervalle
if empty(L) then
    pfad ← pfad ∘ [z]                                   Direktverbindung ist frei
    ok ← true
else if rek ≤ 0 then
    ok ← false                                          Rekursionstiefe erschöpft
else
    via ← viaPoints(a, z, L)                            Liste möglicher Umwegpunkte
    pfad′ ← pfad                                        bisherigen Pfad merken
    ok ← false                                          vorläufig
    while ¬ok ∧ ¬empty(via) do
        q ← head(via)                                   ersten Kandidat ...
        via ← tail(via)                                 ... aus Liste entfernen
        pfad ← pfad′                                    stets vom selben Stand ausgehen
        ok ← connect(a, q, rek − 1, pfad)               rekursive Lösung ...
            ∧ connect(q, z, rek − 1, pfad)              ... der Teilprobleme

return ok
```

Abbildung 1.10: Systematisches Ausprobieren von Umwegpunkten

einer erfolgreichen Berechnung der nichtdeterministischen Maschine konzentriert sich auf derartige Rekursionsbäume beschränkter Tiefe für das gestellte Wegesuchproblem.

Diese Beschränkung birgt allerdings die Gefahr, daß sämtliche Lösungswege von der Konkurrenz ausgeschlossen bleiben, wenn die maximal erlaubte Rekursionstiefe zu klein gewählt wird. Eine sehr große Tiefe führt hingegen dazu, daß nichtterminierende Rechnungen sehr lange verfolgt werden, ehe sie für gescheitert erklärt werden. Dadurch wiederum wird viel Zeit verschwendet, obwohl eventuell einfache Lösungen existieren, die aber erst danach ausprobiert werden. Man ist also bei einer effizienten Implementierung daran interessiert, die erlaubte Rekursionstiefe so klein wie möglich und so groß wie nötig zu wählen.

Da die minimale Tiefe unbekannt ist, mit der ein Bewegungsplanungsproblem gelöst werden kann, stellt sich ein Suchproblem in der Menge der natürlichen Zahlen. Dabei ist eine Anfrage, ob der gewählte Wert zu klein ist oder nicht, allerdings sehr teuer, und man wird sich wohl auch mit dem ersten genügend großen Wert zufrieden geben.

Die Anzahl möglicher Bäume einer festen Tiefe hängt wesentlich von der Anzahl der Umwegpunkte ab, welche auf einer Grenzgeraden ausprobiert werden. Wenn jeweils genau k solcher Ausweichpositionen in Erwägung gezogen werden und r die erlaubte Tiefe des Baums bezeichnet, so gilt für die Anzahl $B_k(r)$ der Bäume

$$B_k(0) = 1,\ \ B_k(r+1) = 1 + kB_k(r)^2$$

Man beachte, daß es einen Baum der Tiefe 0 gibt und k Möglichkeiten, ein Planungsproblem in zwei unabhängige Teile zu zerlegen. Nun betrachte man die höchsten Exponenten $e(r)$ von k bei der Entwicklung von $B_k(r)$.

$$\left.\begin{array}{r} e(0) = 0 \\ e(r+1) = 2e(r) + 1 \end{array}\right\} \Rightarrow e(r) = 2^r - 1$$

Damit ergibt sich weiter

$$B_k(r) = k^{e(r)} + o(k^{e(r)})$$

Das verdeutlicht, warum eine möglichst niedrige Wahl von r von Vorteil ist.

Als Maß für die Laufzeit der Strategie bietet es sich allerdings an, die Anzahl der „intersect“-Aufrufe zu benutzen; diese Aufrufe sind das eigentlich Teure in diesem Verfahren, und nur sie sind direkt von der Komplexität der Hindernisse abhängig. Jedem Knoten des Rekursionsbaums entspricht aufgrund der Struktur unseres Algorithmus ein Schnitt einer Geraden mit dem Hindernis, und zwar für die Direktverbindung. Hat ein Knoten Kinder, so kommt ein weiterer Schnitt für die Grenzgerade hinzu. Wieviele Anfragen an die Unterroutine „intersect“ braucht man nun mindestens, um zu entscheiden, daß ein gegebenes Bewegungsplanungsproblem mit der vorgestellten Strategie nicht lösbar ist? Seien dazu k und r wie zuvor, $I_k(r)$ bezeichne die Anzahl der „intersect“-Aufrufe, welche unser Verfahren verursacht, ehe es **false** zurück gibt. Dann gilt

$$\left.\begin{array}{r} I_k(0) = 1 \\ I_k(r+1) \geq 2 + kI_k(r) \end{array}\right\} \Rightarrow I_k(r) \geq 2\sum_{i=0}^{r-1} k^i + k^r$$

Diese Anzahl ist gleich der unteren Schranke, wenn bei jedem Aufteilen des Problems bereits der erste rekursive Aufruf scheitert, d.h. kein Hinweg zum Umwegpunkt gefunden wird. Andererseits können auf dem Hinweg sehr viele Anfragen verursacht werden, die letztlich sogar einen Weg konstruieren, obwohl anschließend keine Verbindung zum Ziel hergestellt wird. Als obere Schranke ergibt sich allerdings

$$\left.\begin{array}{r} I_k(0) = 1 \\ I_k(r+1) \leq 2 + 2kI_k(r) \end{array}\right\} \Rightarrow I_k(r) \leq 2\sum_{i=0}^{r-1}(2k)^i + (2k)^r \leq 3(2k)^r$$

Das bedeutet aber für die oben erwähnte Suche nach einer geeigneten Rekursionstiefe, daß jedes Ausprobieren eines ungeeigneten r mit $\Omega(k^r)$ Kosten bestraft wird. Das Verhältnis von Ober- zu Untergrenze ist dabei exponentiell in r und läßt einen breiten Spielraum offen.

Implementierte Strategien

Im Rahmen dieser Arbeit wurde das beschriebene heuristische Verfahren zur Wegesuche unter anderem für Bewegungen eines Punktes zwischen Kreisscheiben in der Ebene implementiert. An diesem Programm können grundlegende Tests des Algorithmus vorgenommen und seine Eigenschaften studiert werden. Dabei wurden drei Strategien bereitgestellt, eine geeignete Rekursionstiefe zu bestimmen:

- Die einfachste Vorgehensweise überläßt die Wahl der maximal erlaubten Tiefe dem Benutzer des Programms und dient hauptsächlich interaktiven Versuchen mit dem Algorithmus. Allerdings kann ein erfahrener Benutzer, vor allem wenn gleiche oder ähnliche Szenen wiederholt auftreten, damit rasch gute Ergebnisse erzielen. Häufig sind menschliche Strategien, solche Suchprobleme in Abhängigkeit von Aufgabenstellung und bisheriger Erfahrung zu lösen, auch nur schwer durch ein Programm nachzubilden. Ein vielfach ausreichender Standardwert erleichtert die Benutzung, wenn auch eventuell auf Kosten der Laufzeit.

- Eine schlichte, aber für die betrachteten Szenen durchaus erfolgreiche Strategie besteht darin, mit Rekursionstiefe Null zu beginnen und im Fall des Mißerfolgs diese um Eins zu erhöhen. Dadurch wird auf einfache Weise die minimale Rekursionstiefe bestimmt, mit der sich eine gegebene Szene lösen läßt. Dieses Verfahren ist im allgemeinen schneller als ein Ansatz mit deutlich zu hoher Tiefe, und die gefundenen Wege sind auch eleganter.

- Die obige Methode hat den Nachteil, daß der Teil des Rekursionsbaums in der Nähe der Wurzel immer wieder neu aufgebaut wird, nur um eine tiefere Lage Knoten hinzufügen zu können. Das läßt sich vermeiden, wenn die Ergebnisse früherer Berechnungen gespeichert werden. Dadurch wird es dann auch möglich, verschiedene Umwegpunkte auf derselben Grenzgeraden sowie den jeweiligen Hin- und Rückweg parallel zu verfolgen.

  Die Datenstruktur bildet wieder einen Baum, dessen Knoten Bewegungsplanungsprobleme sind, aber die Kanten entsprechen jetzt paarweise einem bestimmten Umwegpunkt und der dadurch verursachten Aufteilung. Mit obiger Notation erhält man also einen Baum vom Grad $2k$ und Tiefe r, wobei r dynamisch erhöht wird, während der Baum an geeigneten Stellen erweitert wird.

  Sobald Teilprobleme gelöst werden, wird diese Information nach oben propagiert und der entsprechende Unterbaum wird gelöscht; denn wenn das gestellte Problem mit einem der Umwegpunkte gelöst wurde, so können die anderen Ausweichpositionen auf derselben Grenzgeraden vernachlässigt werden. Dadurch wird jeweils der Punkt für den Grenzübertritt verwendet, der die geringste Rekursionstiefe für die Verbindung mit Start und Ziel benötigt, was zu sehr eleganten Wegen führt. Zudem terminiert das Verfahren mit der kleinsten Tiefe, die eine Lösung gestattet, bezogen natürlich auf die Strategie zur Auswahl der Umwegpunkte.

1.8 Wahl der Umwegpunkte

Bei der Beschreibung einer deterministischen Wegesuchstrategie in Abbildung 1.10, Seite 21, wurde eine Funktion „viaPoints($\mathbf{a}, \mathbf{z}, L$)“ verwendet. Zu gegebenem Ausgangs- und Zielpunkt $\mathbf{a}$ und $\mathbf{z}$ sowie einer bereits errechneten Liste L von Intervallen, welche den Schnitt des Liniensegments $(\mathbf{a}, \mathbf{z})$ mit dem Hindernis H repräsentieren, sollen Umwegpunkte bestimmt werden. Das Verfahren dazu wurde im Prinzip bereits geschildert: Eine Grenzgerade senkrecht zum Weg wird mit dem Hindernis geschnitten, in freien Abschnitten darauf ergeben sich die Ausweichpositionen.

Die Wahl der Grenzgeraden ist dabei nicht von entscheidender Bedeutung für die Lösbarkeit eines Problems, wohl aber für die dabei entstehenden Kosten in Form von „intersect“-Aufrufen. Statistiken von Pseudozufallsexperimenten mit der implementierten zweidimensionalen Strategie weisen auf zweierlei hin:

1. Mittelsenkrechten auf einem Abschnitt von $(\mathbf{a}, \mathbf{z}) \cap H$ sind zu bevorzugen. Speziell bei Kreisen sind dies natürlich die günstigen Geraden, um Punkte seitlich des Hindernisses aufzuspüren. Aber auch allgemein gilt wohl, daß eine Gerade sehr nahe am Rand eines Hindernisabschnitts den jenseits gelegenen Punkt benachteiligt und eine Lösung erschwert. Eine Symmetrie bezüglich der lokalen Information erreicht man eben durch ein Lot auf deren Mittelpunkt. In jedem Fall sollte die Grenzgerade nicht auf einem freien Abschnitt von $(\mathbf{a}, \mathbf{z})$ errichtet werden, da dann kein wirklicher Anhaltspunkt zum Umgehen des Hindernisses gegeben wird.

2. Um das Divide-and-Conquer-Konzept auszunutzen, sollte die Grenzgerade auch das ursprüngliche Wegstück $(\mathbf{a}, \mathbf{z})$ gleichmäßig teilen. Dadurch werden die vorhandenen Hindernisse zumindest statistisch gleichmäßig auf Hin- und Rückweg verteilt. Falls eines der beiden Teilprobleme deutlich leichter zu lösen ist, so bestimmt das andere die nötige Rekursionstiefe, und diese kann durch eine fairere Aufgabenteilung verringert werden. Da die Kosten exponentiell mit der Rekursionstiefe wachsen, ergibt sich daraus ein deutlicher Gewinn.

Eine Synthese dieser Beobachtungen und zugleich die im Versuch erfolgreichste Strategie ist das Errichten der Mittelsenkrechten auf dem mittleren Hindernis, d.h. möglichst auf halbem Weg von $\mathbf{a}$ nach $\mathbf{z}$.

Für die Wahl eines Umwegpunktes innerhalb eines Freiintervalls gibt es eigentlich nur eine Möglichkeit, nämlich den Mittelpunkt, wofür sowohl die Experimente als auch die Überlegung der Symmetrie und des maximalen Sicherheitsabstands sprechen.

Weniger eindeutig dagegen ist die Frage, in welcher Reihenfolge die Umwegpunkte vom Verfahren ausprobiert werden sollten. Dies hat offensichtlich einen Einfluß auf die Laufzeit, sofern es gelingt, ungünstige Punkte nicht oder erst zuletzt auszuprobieren. Dabei kann natürlich nicht entschieden werden, welche Ausweichpositionen in einer anderen Zusammenhangskomponente des Freiraums liegen, aber es können doch gewisse Kriterien für leicht erreichbare Punkte angeführt werden. Liegt ein Punkt nahe des direkten Weges und innerhalb eines großen Freiraums auf der ihn bestimmenden Grenzgeraden, so ist er als günstig zu bewerten. Weit entfernte Punkte ergeben oft unnötig lange Pfade, auf dem Weg dorthin sind mehr Hindernisse zu erwarten, und je nach Lokalität der Szene ist die Wahrscheinlichkeit höher, daß sie in einer anderen Zusammenhangskomponenten liegen.

In der Implementierung werden daher zu jedem Umwegpunkt sein Abstand a von der Direktverbindung $(\mathbf{a}, \mathbf{z})$ und die Breite b des ihn umgebenden Freiraums, gemessen auf seiner Grenzgeraden, bestimmt. Als Bewertungsfunktion dient die Abbildung

$$B_\beta : \mathbb{R}_0^+ \times \mathbb{R}^+ \to \mathbb{R}_0^+, \ (a, b) \mapsto \frac{a}{b^\beta}$$

Durch Variation von β lassen sich verschiedene Effekte erzielen; für $\beta = 0$ z.B. hängt die Bewertung nur vom Abstand des Umwegpunktes ab. Gemäß B_β werden die auf einer Grenzgeraden gefundenen Ausweichpositionen nun nach steigendem Funktionswert sortiert, anschließend wird diese Liste auf die besten k Punkte beschränkt. So werden vielversprechende Punkte zuerst versucht und für $k < \infty$ die Zahl der Alternativen begrenzt, die bei der Aufteilung eines Planungsproblems berücksichtigt werden.

Man kann jetzt versuchen, ein optimales β zu bestimmen in dem Sinn, daß eine gewisse Klasse von Problemen leicht gelöst wird. Für die bereits angesprochenen Pseudozufallsexperimente stellte sich heraus, daß eine Berücksichtigung von b deutliche Vorteile bringt; $\beta = 2/3$ war der beste der getesteten Werte. Da $0 \leq \beta < 1$ den Abstand und $1 < \beta$ die Intervallbreite stärker gewichtet, kann hier auch eine Wahl zwischen kurzen Wegen und kurzen Laufzeiten getroffen werden; eine sorgfältige Balance zahlt sich dabei aus.

1.9 Parallelisierbarkeit

Nachdem die Idee der Wegesuchstrategie mitsamt allen wesentlichen Details in der Ebene beschrieben ist, soll noch ein weiterer interessanter Aspekt angesprochen werden, der sich ebenfalls problemlos auf höhere Dimensionen übertragen läßt. Das Verfahren bietet sich aufgrund seines Divide-and-Conquer-Ansatzes geradezu zur Parallelisierung an, wobei sich zwei Ansätze abzeichnen.

Eine Reihe von Prozessoren habe jeweils Zugriff auf die im allgemeinen implizite Beschreibung des Hindernisses; im Fall von verteiltem Speicher muß diese beim Programmstart also einmalig an alle Prozessoren gesandt werden. Dann startet das eigentliche Verfahren mit dem gestellten Problem und einem Prozessor, welcher die Direktverbindung testet und gegebenenfalls Umwegpunkte errechnet. Erfolg oder Scheitern durch mangelnde Rekursionstiefe werden sofort zurück gemeldet. Das Aufteilen des Problems erfolgt auf zwei freie Prozessoren, denen nur Ausgangs- und Zielpunkt mitgeteilt werden muß; Rückmeldungen sind ein boolscher Wert und im Erfolgsfall eine Liste von Punkten, über die der konstruierte Pfad verläuft. Während ein Prozessor auf Rückmeldungen wartet, kann er selbst ein Teilproblem übernehmen, sowohl als Teil des ihm übertragenen Problems als auch im Auftrag eines anderen.

Da das eigentlich rechenintensive eines solchen Bewegungsplanungsverfahrens allerdings der Schnitt eines Liniensegments mit dem Konfigurationshindernis ist, kann auch eine asymmetrische Rollenverteilung erfolgen. Ein Prozessor übernimmt die Ablaufsteuerung und erzeugt Anfragen an die Routine „intersect", welche von einer Reihe von Prozessoren bearbeitet werden. Durch Verwendung einer Baumstruktur mit Informationen über das Planungsproblem und seine Teile, wie sie zum Auffinden der minimalen Rekursionstiefe geschildert wurde, lassen sich rasch Anfragen generieren. Diese werden parallel gelöst und bei Gelegenheit verwendet, um den Baum zu erweitern; der Master-Prozessor sollte mit der Verwaltung der Information ausreichend beschäftigt sein. Die Arbeitsverteilung seitens der untergeordneten Prozessoren gestaltet sich einfacher: Aufträge kommen, werden am Stück abgearbeitet und beendet.

Je nach Art der Information über das Hindernis und abhängig von der konkreten Aufgabe läßt sich hier auch eine Spezialisierung einzelner Prozessoren vornehmen. Bei mehreren Arten von Freiheitsgraden für Bewegungen kann z.B. je ein Prozessor Translationen oder Rotationen behandeln, bei einem mehrgliedrigen Roboter z.B. je ein Gelenk. Die Ergebnisse der Teilrechnungen können dann nach Bedarf kombiniert werden.

Mit diesen Bemerkungen schließt das erste Kapitel und wir wenden uns alsbald höheren Dimensionen zu.

Kapitel 2

Übertragung auf $\mathbb{R}^n$

Bis jetzt wurde die heuristische Bewegungsplanungsstrategie nur für einen Punkt beschrieben, der sich im $\mathbb{R}^2$ bewegt. Dies erlaubte eine anschauliche Erläuterung der Idee und einfache zeichnerische Darstellungen. Neben dem Grundgedanken des Verfahrens wurden auch schon einige Probleme deutlich, so zum Beispiel die Frage nach der Terminierung und Vollständigkeit. Ferner konnte die Wahl und Bewertung von Umwegpunkten eingeführt und insbesondere eine geeignete Rekursionsstrategie untersucht werden. Viele experimentelle statistische Analysen des Verfahrens lassen sich im $\mathbb{R}^2$ besonders leicht und schnell vornehmen.

In der praktischen Arbeit stand daher auch die Implementierung der zweidimensionalen Wegesuche am Anfang. Da das Ziel letztlich aber die Bewegung von Körpern im Raum war, mußte der Schritt in höhere Dimensionen erfolgen. Dazu wurde das bewegte Objekt zunächst wieder auf einen Punkt beschränkt, der allerdings zwischen Polyedern im Raum navigierte.

Es ist jedoch nicht nur historisch bedingt, sondern auch didaktisch sinnvoll, zunächst nur Bewegungen für einen Punkt in höherdimensionalen Räumen zu betrachten. Dabei können sowohl Probleme bei der Suche nach und Auswahl von Umwegpunkten studiert werden als auch der Gedanke, gewisse Einschränkungen hinsichtlich der erlaubten Richtungen eines Weges vorzunehmen. Später wird dann geschildert, wie sich durch das Konzept des Konfigurationsraums auch das Problem, einen Körper kollisionsfrei zu verschieben und sogar zu drehen, zurückführen läßt auf die Wegesuche für einen Punkt. Dazu ist vor allem die Unterroutine „intersect“ an die konkreten Gegebenheiten anzupassen, die Planungsstrategie selbst bleibt erhalten.

Eine solche Übertragung ist für viele konkrete Probleme der Bewegungsplanung möglich, erfordert jedoch einige problemspezifische Erweiterungen der Strategie. Insofern werden in diesem Kapitel ganz allgemeine Ideen untersucht, während danach sozusagen exemplarisch die Anwendung auf bestimmte interessant scheinende Spezialfälle beschrieben wird. Dies mag auch als Anregung dienen, wie zum Beispiel Bewegungen für einen Roboterarm oder mehrere unabhängige Körper geplant werden können. Viele der im folgenden beschriebenen Algorithmen können zur Lösung solcher Probleme verwendet werden, darauf wird auch an gegebener Stelle verwiesen.

2.1 Besonderheiten bei der Suche nach Umwegpunkten

Betrachten wir als Problemstellung wieder ein Hindernis $H \subset \mathbb{R}^n$ und zwei Punkte $\mathbf{a}, \mathbf{z} \in \mathbb{R}^n \setminus H$ außerhalb davon, einen Anfang und ein Ziel. Mit der bekannten Divide-and-Conquer-Strategie soll ein kollisionsfreier Weg gesucht werden, d.h. eine stetige Kurve im Raum von $\mathbf{a}$ nach $\mathbf{z}$, die nicht durch H verläuft. Dazu wird wieder die direkte Verbindung $l_{\mathbf{a},\mathbf{z}}$ betrachtet und deren Schnitt mit H. Falls sich eine Kollision findet, so soll z.B. vom Mittelpunkt $\mathbf{p} \in l_{\mathbf{a},\mathbf{z}}$ des ersten Kollisionsintervalls aus senkrecht zu $\mathbf{z} - \mathbf{a}$ ausgewichen werden.

In der Ebene ergab sich dadurch eindeutig eine Gerade, auf der mögliche Umwegpunkte liegen können. Jetzt aber, mit mehr Dimensionen, ist lediglich eine Hyperebene beschrieben, in der man nach ihnen suchen muß. Wir wollen jedoch nicht etwa versuchen, den Schnitt dieser Hyperebene mit H zu konstruieren, dies wäre im allgemeinen nicht praktikabel. Vielmehr beschränken wir uns weiterhin darauf, Anfragen nach dem Schnitt einer Gerade bzw. eines Liniensegments mit dem Hindernis zu stellen.

Damit stellt sich das Problem, geeignete Geraden in der Hyperebene zu wählen, um auf diesen wiederum in bekannter Weise Ausweichmöglichkeiten zu finden. Bei einer beschränkten Hindernismenge enthält natürlich jede Gerade freie Punkte, allerdings müssen diese nicht in derselben Zusammenhangskomponente von $\mathbb{R}^n \setminus H$ liegen wie $\mathbf{a}$. Ähnlich wie in der Ebene wird man also versuchen, eine Reihe von Kandidaten für Umwegpunkte zu bestimmen, welche dann durch eine geschickte Rekursionsstrategie ausprobiert werden. Auf verschiedene Möglichkeiten zur Wahl der Geraden sowie ihre Vor- und Nachteile soll nun noch näher eingegangen werden.

2.1.1 Zwei fest gewählte Geraden

Durch die Bewegungsrichtung $\mathbf{z}-\mathbf{a}$ ist ein Vektor vorgegeben, den man durch $\mathbf{r}_2, \ldots, \mathbf{r}_n$ zu einer Orthogonalbasis des $\mathbb{R}^n$ ergänzt. Dabei hat man natürlich noch gewisse Freiheiten, nämlich z.B. für $n = 3$ einen Drehwinkel um die Achse $l_{\mathbf{a},\mathbf{z}}$. Diese sollen hier aber zunächst irgendwie deterministisch von $\mathbf{a}$ und $\mathbf{z}$ abhängen und interessieren erst später wieder. Zum Auffinden von Umwegpunkten werden die Geraden durch $\mathbf{p}$ mit Richtung $\mathbf{r}_i$ betrachtet, freie Intervalle darauf liefern Kandidaten für Umwegpunkte, z.B. als Intervallmitte.

Ein Vorteil davon ist sicher der Determinismus der Methode, insofern als man reproduzierbare Ergebnisse erhält; die Güte der Lösung ist auch nicht vom Zufall abhängig, eventuell aber konstant schlecht. Ferner scheint eine solche Basis irgendwie ausgezeichnet vor beliebigen Richtungen, da sie auch alle Raumdimensionen gleichmäßig ausnutzt; in Szenen, deren Hindernisse sich an den Achsen des Koordinatensystems orientieren, bieten so geplante Wege auch oft elegante Lösungen. Der Rechenaufwand ist ebenfalls gering und die Aussicht hoch, genügend Kandidaten zu finden, zumal sich schon auf einer Geraden oft mehr Kandidaten finden, als die Rekursionsstrategie ausprobieren kann bzw. will.

Ein klarer Nachteil tritt zutage, falls kein Umwegpunkt zu einer Lösung führt; dann helfen auch erneute Planungsversuche nicht. Zudem sollte man Vorsorge treffen, falls doch einmal gar keine oder zuwenig Kandidaten gefunden werden; dann sollte mit einer anderen Methode nach weiteren Ausweichgelegenheiten gesucht werden. [Sch92] schlägt für diesen Fall vor, die Freiheitsgrade zu nutzen und die Basis z.B. um $\mathbf{z}-\mathbf{a}$ zu drehen; dafür lassen sich auch aufwendige Strategien denken, die jedoch selten zum Einsatz kommen werden.

Eine spezielle Möglichkeit bietet sich an, um z.B. nur achsenparallele Wege zu planen. In diesem Fall erlaubt man nur einen festen Satz von Richtungen, in denen Ausweichbewegungen erlaubt sind. Es muß aber darauf geachtet werden, daß auch die „direkte Verbindung“ von $\mathbf{a}$ und $\mathbf{z}$ abschnittsweise nur in solche Richtungen verläuft. Ferner müssen zu jedem Vektor auch einige dazu senkrechte erlaubt sein. Wir werden später im Abschnitt 2.2 eine Strategie kennenlernen, die solche Einschränkungen der Bewegungsrichtungen in einer speziellen Weise gestattet.

2.1.2 Zufällige Geraden

Alternativ würfele man eine Anzahl k von Richtungen $\mathbf{r}_1, \ldots, \mathbf{r}_k \in \mathbb{R}^n$, $\forall 1 \leq i \leq k: \ \mathbf{r}_i \perp (\mathbf{z}-\mathbf{a})$. Dadurch ergeben sich k Geraden durch $\mathbf{p}$, die alle in der Hyperebene liegen. Die Anzahl k sollte im allgemeinen mit der Dimension des Raumes ungefähr übereinstimmen, dann kann man im Mittel erwarten, daß die Richtungen die Raumdimensionen gleichmäßig abdecken.

Das Verhalten sollte ähnlich zu einer festen Basis sein, mit dem Nachteil fehlender Beständigkeit und dem Vorteil, daß Fehlschläge nicht permanent sind. Es erweist sich allerdings in praktischen Beispielen, daß Hindernisse oft rechtwinklig aufgebaut und an den Koordinatenachsen orientiert sind, man denke z.B. an Wände, Tische usw. Daher haben zufällige Richtungen wenig mit diesen Problemen zu tun und erschweren die Lösung eher, was aber bei anderen Anwendungen vielleicht weniger ins Gewicht fällt. Um die Anzahl der gefundenen Kandidaten zu regulieren, läßt sich der Wert k natürlich dynamisch bestimmen, d.h. man betrachtet solange neue Geraden, bis genügend aussichtsreiche Umwegpunkte feststehen.

2.1.3 Allgemeine Lage der Geraden in der Hyperebene

Man muß sich nicht nur auf Geraden durch $\mathbf{p}$ beschränken, sondern kann diese durchaus beliebig innerhalb der Hyperebene wählen. Zur Auswahl der Richtungen lassen sich die oben genannten Strategien verwenden. Interessant ist aber, wie man den Aufpunkt der Geraden in der Hyperebene bestimmt bzw. wie man diese Wahl anschaulich deuten kann.

Die Verschiebung des Aufpunkts, nennen wir ihn $\mathbf{q}$, gegenüber $\mathbf{p}$ kann als Folge einer Reihe von Bewegungen erklärt werden, deren Richtung jeweils mit einem der genannten Verfahren bestimmt wurde. Für jede solche Richtung wird eine zufällige Schrittweite gewählt. Es wird allerdings keine Anfrage nach dem Schnitt der entsprechenden Geraden mit dem Hindernis gestellt; vielmehr wird die Bewegung ohne Rücksicht auf das Hindernis in Gedanken ausgeführt. Auf diese Weise wirken verschiedene Richtungsvektoren zusammen bei der Bestimmung des Aufpunkts, obwohl nur ein einzelner in der Orientierung der Grenzgeraden sich wiederfindet.

Welchen Vorteil hat nun diese Vorgehensweise? In späteren Varianten des Verfahrens zur heuristischen Wegesuche werden mehrere unabhängige Arten von Ausweichbewegungen möglich sein, von denen aber jeweils nur eine durch eine „intersect"-Anfrage behandelt werden kann. Um trotzdem mehrere Bewegungsarten bei der Suche nach Umwegpunkten zu kombinieren, ergibt sich allgemein folgende Vorgehensweise. Man wählt mehrere erlaubte Ausweichrichtungen aus und für alle eine Schrittweite; dann sei $\mathbf{q}$ der Punkt, der aus $\mathbf{p}$ durch die gewählten Schritte in die jeweiligen Richtungen hervorgeht. Von $\mathbf{q}$ aus wird dann eine lineare Bewegung auf ihre Kollisionen geprüft, d.h. es wird eine Anfrage nach dem Schnitt einer Geraden innerhalb der Hyperebene durch $\mathbf{q}$ mit dem Hindernis H gestellt. Diese letzte Bewegungsrichtung wählt man ebenfalls mit einer der bekannten Strategien.

Die Schrittweite für jede Richtung wird man wohl zufällig wählen, wobei kleine Entfernungen zu bevorzugen sind, damit die Umwegpunkte nicht zu weit vom Kurs abkommen. Eine entscheidende Rolle kommt dabei auch dem Zusammenspiel mit der Bewertung der Ausweichpunkte zu.

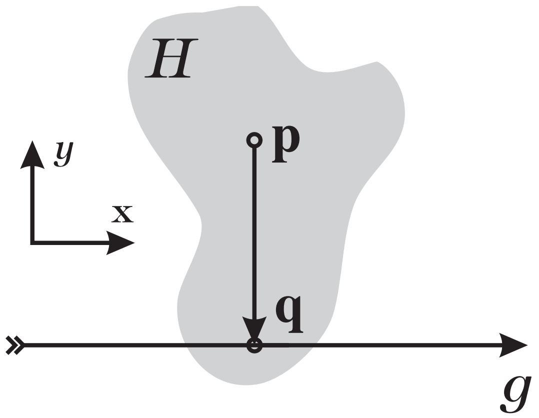


Abbildung 2.1: Grenzgerade mit Aufpunkt ausserhalb von $l_{\mathbf{a},\mathbf{z}}$

Durch Abbildung 2.1 soll diese Situation veranschaulicht werden; dabei ist eine Bewegung für einen Punkt im $\mathbb{R}^3$ gesucht. Die Direktverbindung von Ausgangs- und Zielpunkt war nicht kollisionsfrei, und daher wurde senkrecht dazu eine Hyperebene betrachtet; diese ist hier dargestellt. Nehmen wir weiter an, daß nicht alle Raumrichtungen für eine Bewegung erlaubt seien, sondern lediglich die der Koordinatenachsen; im Abschnitt 2.2 findet sich ein Planungsstrategie dafür. Innerhalb der Hyperebene verbleiben davon noch die x- und y-Richtung, wobei eine Ausweichbewegung in y-Richtung samt negativer Schrittweite ausgewählt wird. Diese Bewegung verschiebt den bisherigen Aufpunkt $\mathbf{p}$ der Grenzgeraden auf den Punkt $\mathbf{q}$; die Richtung der Grenzgeraden ergibt sich z.B. als die der x-Achse. So wirken unabhängige Bewegungsrichtungen zusammen und ergeben eine Grenzgerade, bezüglich derer ein „intersect"-Aufruf erfolgt.

2.1.4 Auswahl von Umwegpunkten

Bereits bei der Beschreibung der heuristischen Wegesuchstrategie in der Ebene wurde erläutert, welche Bedeutung der Bewertung und Auswahl von Umwegpunkten hinsichtlich des Erfolgs der Bewegungsplanung zukommt. Dort wird auch eine Klasse von Wertungsfunktionen vorgestellt und diskutiert. In höherdimensionalen Räumen ergeben sich im allgemeinen noch mehr Kandidaten für die Umwegpunkte; ein effizientes Durchsuchen des Rekursionsbaum erfordert aber, daß jeweils nur einige wenige Alternativen ausprobiert werden. Daher wird man wieder nur die besten Ausweichmöglichkeiten — gemäß einer Gütefunktion — in die Liste der Umwegpunkte aufnehmen, über die eine Verbindung von Anfang und Ziel versucht wird. Es ist allerdings sinnvoll, zumindest hat sich das in praktischen Beispielen bei Rotationen und Translationen gezeigt, daß man die verschiedenen Bewegungsarten bzw. Raumrichtungen gleichmäßig berücksichtigt. Mit anderen Worten, selbst wenn alle Ausweichpunkte durch Translation besser sind als die rotatorischen, sollte doch ein Umwegpunkt aufgrund einer Drehung ausprobiert werden.

2.2 Wegeplanung für getrennte Freiheitsgrade

Nach dem bisher Gesagten ist klar, wie die heuristische Strategie zur Wegesuche in einem höherdimensionalen Raum aussieht. Dabei wurde hauptsächlich auf einen Punkt Bezug genommen, der sich im $\mathbb{R}^n$ bewegte, einem Raum, der sozusagen homogen hinsichtlich der Koordinaten ist. Nun soll untersucht werden, wie gewisse Einschränkungen an die jeweilige Richtung des geplanten Weges vorgenommen werden können, indem nämlich nicht mehr alle Koordinaten gleich behandelt werden. Dies hat seinen Ursprung darin, daß bei konkreten Anwendungen z.B. auf einen frei im Raum beweglichen Körper sowohl Ort als auch Orientierung festgelegt werden, zwei im Grunde wesensfremde Koordinaten also. Daher erweist es sich als günstig, diese auch spezifisch zu behandeln und einen Weg aus reinen Translationen und Rotationen zu komponieren.

Der Konfigurationsraum

$$K = K_0 \times \cdots \times K_{n-1} \subseteq \mathbb{R}^d$$

soll jetzt als n-faches kartesisches Produkt angesehen werden, wodurch die d verfügbaren Freiheitsgrade in n Gruppen eingeteilt werden. Dies kann man so deuten, daß Punkte des d-dimensionalen Konfigurationsraums durch n Koordinaten beschrieben werden, die jedoch nicht notwendig reelle Zahlen, sondern eventuell wiederum Vektoren sind. Wege in diesem Raum müssen stückweise in den einzelnen Teilräumen verlaufen, ändern also jeweils nur eine Koordinate.

Für die Planungsebene ist es gleichgültig, was die einzelnen Koordinaten wirklich bedeuten; lediglich die Funktionen zur Kollisionsrechnung und zum Finden von Umwegpunkten interpretieren diese Werte. Eine solche Aufteilung der Freiheitsgrade kann die

Arbeit der Kollisionsrechnung stark vereinfachen, da nur die n einzelnen „Elementarbewegungen“ und nicht auch deren Überlagerungen auftreten. Bei Roboterarmen mit mehreren Gelenken entspricht dies dem Bewegen derselben nacheinander, wobei Gelenke mit mehr als einem Freiheitsgrad auch als eine Einheit behandelt werden können; das ist eine durchaus gängige Einschränkung der erlaubten Bewegungen.

Nun zur Idee des Algorithmus: Ausgehend von der Anfangslage wird ein Pfad zum Ziel aufgebaut. Als direkte Verbindung von Anfang und Ziel wird ein Weg angesehen, der reihum jede Koordinate verändert; dabei wird Buch geführt, welche Koordinate jeweils zuerst geändert werden soll. Diese Beschränkung ist beim Aufruf vielleicht störend, sorgt aber später dafür, daß bereits bekannte Kollisionen auf dem Weg über einen Umwegpunkt zum Ziel vermieden werden.

Um das besser zu verstehen, betrachte man zwei Punkte $\mathbf{a}$ und $\mathbf{z}$ in Abbildung 2.2, die sich nur in einer Koordinate unterscheiden; versucht man jetzt auf dem Weg von $\mathbf{a}$ zum Ausweichpunkt $\mathbf{q}$ zuerst, auf $\mathbf{z}$ zuzugehen, trifft man natürlich wieder auf das bekannte Stück des Hindernisses, welches den Weg von $\mathbf{a}$ nach $\mathbf{z}$ versperrt. Daher ist es wichtig, die Koordinatenänderung, die der Ausweichbewegung entspricht, vor jener vorzunehmen, welche die errechnete Kollision hervorrief.

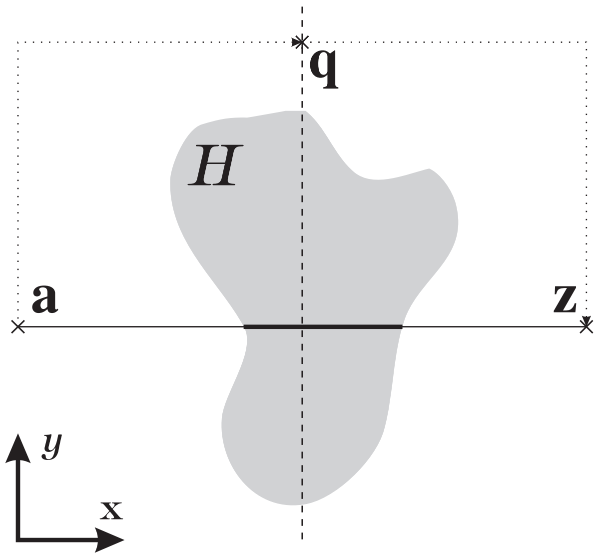


Abbildung 2.2: Reihenfolgebedingung beim Ausweichen

Die zyklische Vorgehensweise beim Verändern der Koordinaten genügt dieser Reihenfolgebedingung und läßt sich einfach durchführen. Gerade in höheren Dimensionen ist es ausgeschlossen, alle Permutationen der Koordinaten durchzuprobieren, um eine Direktverbindung zu finden; daher sollte man sich auf eine geeignete Abfolge festlegen. Falls die betrachtete Reihenfolge in einer konkreten Szene ungeschickt ist, so führt das lediglich zu vermehrten Ausweichbewegungen.

Tritt auf einem Teilstück des direkten Wegs eine Kollision auf, so kann über Umwegpunkte ausgewichen werden, die orthogonal zur kollidierenden Bewegung liegen.

Orthogonal heißen Bewegungen in anderen Teilräumen oder solche im selben Teilraum, deren Richtung orthogonal zur fraglichen Bewegungsrichtung ist. Über die besonderen Probleme in höheren Dimensionen, solche Richtungen zu bestimmen, wurde bereits berichtet; die entsprechenden Verfahren lassen sich hier sinngemäß verwenden. Die in unserem Verfahren grundlegende Idee des senkrechten Ausweichens ist dabei lediglich geeignet zu interpretieren.

geg.:

$K = K_0 \times \cdots \times K_{n-1} \subseteq \mathbb{R}^d$	Konfigurationsraum
$a = (a_0, \ldots, a_{n-1}) \in K$	Anfang
$z = (z_0, \ldots, z_{n-1}) \in K$	Ziel

Aufruf:

connect$(a, z, 0, [a])$	$[a]$ bezeichne einelementige Liste

Bemerkung 2.1:
In dieser Strategie wird zum Ausweichen nur ein einziger Umwegpunkt q errechnet. Dies ist stellvertretend für ein Ausprobieren vieler Punkte oder im Sinne einer nichtdeterministischen Maschine zu verstehen, die solche Punkte geeignet rät.

connect$(a, z : K;\ j : 0 \ldots n-1;\ pfad : \text{List}(K)\ ) : \text{List}(K)$

$i, k : \mathbb{N}$
$p, q : K$
L : IntervalList

$i \leftarrow j$ — zu ändernde Koordinate
$p \leftarrow a$ — aktuelle Position (wandert von a nach z)

repeat
  $p_i \leftarrow z_i$
  $L \leftarrow$ intersectSegment(a, p) — Kollisionsintervalle
  if empty(L) **then**
    $pfad \leftarrow pfad \circ [p]$
    $a \leftarrow p$
    $i \leftarrow i + 1 \pmod n$
until $\neg$ empty$(L) \vee a = z$

if $\neg$ empty(L) **then**
  $(q, k) \leftarrow$ viaPoints(a, p, L) — q weicht in Koordinate k aus
  $pfad \leftarrow$ connect$(a, q, k, pfad)$ — k vor i ändern!
  $pfad \leftarrow$ connect$(q, z, i, pfad)$ — i vor k ändern!

return $pfad$

Abbildung 2.3: Planungsheuristik für getrennte Freiheitsgrade

Kapitel 3

Ein Körper im $\mathbb{R}^3$

Wir haben gesehen, wie ein Punkt Hindernissen im $\mathbb{R}^3$ ausweichen kann. Wenn jetzt anstelle des Punktes ein Körper im Raum bewegt werden soll, so läßt sich dessen Bewegung auf die eines Punktes in einem *Konfigurationsraum* zurückführen, wie in Kürze gezeigt wird. Dieser Begriff wurde zuvor im Zusammenhang mit Punkten bereits gebraucht, doch erst jetzt erhält er seine wahre Bedeutung. Für einen einzelnen starren Körper mit translatorischen Freiheitsgraden im Raum wird sich dieser Konfigurationsraum als $\mathbb{R}^3$ herausstellen; d.h. das Problem, einen Körper zu bewegen, kann auf die Bewegung eines Punktes im $\mathbb{R}^3$ reduziert werden.

3.1 Das Konzept des Konfigurationsraums

Parametrisierung der Lage Zuerst muß geklärt werden, wie die Lage eines im Raum frei verschiebbaren Körpers, der immer die gleiche Orientierung beibehält, festgelegt und durch geeignete Parameter beschrieben werden kann. Faßt man den dreidimensionalen euklidischen Anschauungsraum auf als affinen Raum über dem Vektorraum $\mathbb{R}^3$, so entsprechen die Verschiebungen eines starren Körpers gerade den Abbildungen

$$T_{\mathbf{v}} : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{x} \mapsto T_{\mathbf{v}}(\mathbf{x}) := \mathbf{x} + \mathbf{v}$$

wobei $\mathbf{v} \in \mathbb{R}^3$ Translationsvektor heißt. Falls der starre Körper in irgendeiner Ausgangslage durch eine Punktmenge $P \subset \mathbb{R}^3$ beschrieben ist, so verstehe man unter dem Körper P in Lage $\mathbf{v}$ gerade $T_{\mathbf{v}}(P)$, die um $\mathbf{v}$ verschobene Punktmenge. Damit wird die ausgewählte Startposition durch den Nullvektor $\mathbf{0}$ bezeichnet.

Wegen der Dimension des Anschauungsraums ist klar, daß drei reelle Parameter zur eindeutigen Beschreibung einer Verschiebung notwendig sind. Hieraus ist unmittelbar zu ersehen, daß $\mathbf{v}$ eine geeignete Parametrisierung der Lage des Körpers darstellt, d.h. ohne Redundanz und doch leicht zu handhaben; die Menge aller Positionen ergibt sich zu $\mathbb{R}^3$.

Der Konfigurationsraumansatz Eine auf dem Gebiet der automatischen Bewegungsplanung weitverbreitete und nützliche Idee ist es nun, nicht den Körper in seiner

Hindernisumgebung, dem *Objektraum*, zu betrachten, sondern vielmehr die Menge aller möglichen Lagen des Körpers, seinen *Konfigurationsraum*, zu bilden und diese in erlaubte und verbotene Positionen zu unterteilen. Verbotene oder *unfreie* Plazierungen eines Körpers sind solche, bei denen er im Objektraum mit einem der Hindernisse einen inneren Punkt gemeinsam hat. Positionen, bei denen sich die Oberflächen und nur diese berühren, heißen *halbfrei*; alle anderen *frei*.

Ein Weg für ein Objekt wird stets unfreie Positionen meiden und darf immer über freie verlaufen, halbfreie Positionen sind je nach Anwendung erlaubt oder unerwünscht. Um z.B. kürzeste Wege zu berechnen, sind halbfreie Plazierungen unverzichtbar, damit die Menge der erlaubten Positionen abgeschlossen ist. Interessiert man sich dagegen für sichere Wege, bei denen der bewegte Körper Abstand hält zu seiner Umgebung, dann wird man auf halbfreie Lagen verzichten.

Bei dieser Konstruktion entsprechen die ursprünglichen Objektraumhindernisse gewissen Konfigurationsraumhindernissen, deren Aussehen auch vom zu bewegenden Objekt abhängt. Es gibt etliche Ansätze, solche transformierten Hindernisse und damit eine vollständige Beschreibung des Konfigurationsraums zu errechnen, doch sind alle Verfahren exponentiell in der Dimension des Raums. Die Anzahl der zur Beschreibung der Lage eines Körpers notwendigen Parameter nennt man auch seine Freiheitsgrade, sie bestimmen die Dimension des Konfigurationsraums; in dem momentan betrachteten Fall sind es drei. Später werden auch Rotationen zugelassen, wodurch der Körper drei weitere Freiheitsgrade gewinnt, der Konfigurationsraum wird also sechsdimensional sein.

Sechs scheint noch eine kleine Zahl, selbst für einen exponentiellen Algorithmus. Praktische Implementierungen solcher vollständiger Algorithmen zur Bewegungsplanung, die Hindernisse in den Konfigurationsraum transformieren, zeigen allerdings, daß aufgrund des Laufzeit- und Platzbedarfs diese Strategien eigentlich unbrauchbar sind. Die wesentliche Idee in dieser Arbeit, und auch schon in [Sch92], besteht nun darin, das Aussehen der Konfigurationsraumhindernisse nicht explizit zu berechnen und zu speichern. Vielmehr wird eine heuristische Strategie zur Wegesuche verwendet, die nur Anfragen nach dem Schnitt einer Gerade mit diesen Hindernissen stellt; solche Anfragen können leicht beantwortet werden, wie wir gleich sehen. Dabei nutzt man aus, daß eine Gerade im Konfigurationsraum einem Weg des Körpers im Objektraum entspricht, der Schnitt der Gerade mit den transformierten Hindernissen also den freien und unfreien Abschnitten der Bewegung.

3.2 Kollisionserkennung

Ehe wir zur Beschreibung einer heuristischen Strategie für translatorische Freiheitsgrade gelangen, muß noch eine wichtige Grundlage geschaffen werden, nämlich die Erkennung der kollisionsfreien bzw. unfreien Abschnitte einer Translation. Eine solche Verschiebung wird vom Planungsalgorithmus nachgefragt unter Angabe einer Ausgangsposition und eines Translationsvektors. Bekannt sind ferner ein bewegtes Objekt und eine Menge feststehender Hindernisse.

Faßt man nun den Weg des Objekts während der Verschiebung als stetige Abbildung $\rho : [0,1] \rightarrow \mathbb{R}^3$ auf, wobei $\rho(0)$ die Anfangs- und $\rho(1)$ die Endlage ist, so liefert die Kollisionsrechnung jene Zeitpunkte der Bewegung, zu denen das bewegte Objekt mit einem Hindernis kollidiert. Dieses Ergebnis kann man durch eine Folge von Intervallen beschreiben.

Die Frage nach Kollisionen des bewegten Körpers mit den Hindernissen wird zurückgeführt auf ein logisches Prädikat für die Überlappung zweier Körper. Durch eine geeignete Strategie läßt sich nicht nur für einen bestimmten Zeitpunkt der Bewegung der Wahrheitswert dieses Prädikats ermitteln, sondern sogar die Menge aller Zeitpunkte, zu denen es wahr ist. Dazu wird der logische Ausdruck ausgewertet, wobei allerdings nicht nur *wahr* und *falsch*, sondern ganze Intervalle von Zeitpunkten als Ergebnisse verwaltet werden. Dieses Konzept geht zurück auf [Ca87].

3.2.1 Darstellung der Objekte

Um eine effiziente Kollisionsrechnung zu ermöglichen, beschränken wir die erlaubten Körper auf Polyeder. Dadurch werden die zu lösenden Ungleichungen später einen niedrigen Grad haben, denn die Oberfläche eines Polyeders läßt sich einfach beschreiben. Zudem kann man reale Körper durch Polyeder recht gut annähern, wenn auch der Aufwand sehr hoch werden kann. Zunächst soll nun erläutert werden, was unter einem Polyeder zu verstehen ist.

Definition 3.1:
Eine konvexe polyhedrale Region *im* $\mathbb{R}^3$ *ist der Schnitt endlich vieler abgeschlossener Halbräume; durch endliche Vereinigung davon ergibt sich eine* polyhedrale Region*. Ein* Polyeder *schließlich ist eine polyhedrale Region, die beschränkt und zusammenhängend ist sowie gleich dem Abschluß ihres offenen Inneren.*

Diese Definition hat zur Folge, daß ein Polyeder eine kompakte Teilmenge des $\mathbb{R}^3$ ist, begrenzt durch ebene Flächen, die ihrerseits Polygone sind. Es liegt nun zwar nahe, anzugeben wie das Polyeder aus Halbräumen durch Schnitt und Vereinigung hervorgegangen ist, um so das entstandene Volumenobjekt zu beschreiben. Für die weitere algorithmische Verwendung ist es jedoch günstiger, lediglich den Rand des Polyeders, seine Oberfläche, zu betrachten. Eine solche Darstellung wird auch *boundary representation* genannt.

Damit läßt sich ein Polyeder P beschreiben durch die Menge seiner Ecken, Kanten und Flächen sowie deren Nachbarschaftsbeziehungen. Als seine Beschreibungskomplexität $|P|$ bezeichnet man die Anzahl aller Ecken, Kanten und Flächen: $e + k + f$. Nach einem Satz von Euler gilt $e + f = k + 2$ und damit $|P| = 2k + 2 = \Theta(k)$.

Falls eine der Flächen nicht konvex ist, kann sie leicht in eine Menge konvexer Flächen zerlegt werden. Eine Triangulierung eines Polyeders läßt sich z.B. in quasilinearer Zeit bezüglich der Anzahl der Kanten berechnen und erhöht die Komplexität nur unwesentlich. Bei der Aufteilung einer Fläche F mit k_F Kanten entstehen $k_F - 3$ zusätzliche Flächen, daher gilt

$$f' = f + \sum_F (k_F - 3) = f + \sum_F k_F - 3f = 2(k - f) = 2(e - 2)$$

Die Zahl der Ecken ändert sich nicht; die der Kanten bestimmt man wie folgt, denn jede triangulierte Fläche hat drei Randkanten, die zu je zwei Flächen gehören.

$$2k' = \sum_{F'} k'_{F'} = 3f' \quad \Rightarrow \quad k' = 3(k - f)$$

Damit ergibt sich insgesamt $|P'| = e' + k' + f' = e + 3(k - f) + 2(e - 2) \leq 3|P|$. Daher wird von nun an vorausgesetzt, daß der Rand des Polyeders durch konvexe Polygone repräsentiert wird.

Indem man triangulierte Flächen betrachtet, sieht man mit $3f' = 2k'$ auch

$$e' = k' + 2 - f' = 1/3k' + 2 \quad \Rightarrow \quad 6(e' - 2) + 2 = |P'| \geq |P|$$

und wegen $e' = e$ gilt daher $|P| = \Theta(e)$. Berücksichtigt man zusätzlich, daß in jeder Ecke E mindestens drei Kanten münden, so ergibt sich $3e \leq \sum_E k_E = 2k$ und damit

$$f = k + 2 - e \geq 1/3k + 2 \geq 1/6|P|$$

Insgesamt bedeutet dies

$$e, k, f = \Theta(|P|)$$

und deshalb wird im folgenden nur die „Gesamtkomplexität" des Polyeders verwendet.

Welche Anforderungen stellen die im folgenden beschriebenen Algorithmen an eine Datenstruktur „Polyeder"? Zunächst muß es möglich sein, alle Flächen von P in beliebiger Reihenfolge aufzuzählen. Zu jeder Fläche wird eine Normalengleichung der zugehörigen Ebene benötigt, wobei der Normalenvektor $\mathbf{n}$ bzgl. des Polyeders nach außen gerichtet sei; dadurch ergibt sich eine lokale Innen-Außen-Information, die zuweilen nützlich ist. Ferner werden die Eckpunkte jeder Fläche f benötigt, und zwar in einem positiven Drehsinn bzgl. $\mathbf{n}$. Die Anordnung erfolgt also gegen den Uhrzeigersinn, wobei f gegen den Normalenvektor betrachtet wird; diese Ansicht heißt auch Außenseite von f. Von jedem dieser Eckpunkte ist sein Ortsvektor bekannt. Es ist klar, daß diese Möglichkeiten genügen, um z.B. auch die Randkanten einer Fläche zyklisch zu durchlaufen.

3.2.2 Statische Kollisionserkennung

Kommen wir nun zurück zum Test auf Überlappung zweier Polyeder. Eine solche liegt vor, wenn sich die Oberflächen der Polyeder durchdringen oder eines ganz im anderen enthalten ist. Für den ersten Fall wird nun ein logisches Prädikat hergeleitet, der zweite Fall wird später im Algorithmus etwas anders behandelt werden. Man überlegt sich weiter, daß ein Schnitt der Polyederoberflächen genau dann vorliegt, wenn eine Kante eines Polyeders eine Fläche des anderen schneidet; dabei seien auch degenerierte Fälle eingeschlossen.

Seien also der Anfangs- und Endpunkt $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ einer Kante eines Polyeders gegeben, welche als Liniensegment geschrieben wird:

$$l_{\mathbf{p},\mathbf{q}} := \{\mathbf{p} + \lambda(\mathbf{q} - \mathbf{p}) \mid 0 \leq \lambda \leq 1\} \subseteq \mathbb{R}^3$$

Ferner sei eine konvexe Fläche des anderen Polyeders beschrieben durch eine Ebene

$$E_{\mathbf{n},n_0} := \{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{n}^{\mathrm{T}}\mathbf{x} = n_0\} \subseteq \mathbb{R}^3,\ \mathbf{n} \in \mathbb{R}^3,\ |\mathbf{n}| = 1,\ n_0 \in \mathbb{R}$$

mit Normalenvektor $\mathbf{n}$ und Abstand n_0 vom Ursprung und die verschiedenen Eckpunkte

$$\mathbf{v}_0, \ldots, \mathbf{v}_{k-1} \in E_{\mathbf{n},n_0},\ k \geq 3,\ \forall\, 0 \leq i < j < k:\ \mathbf{v}_i \neq \mathbf{v}_j$$

Zur Vereinfachung des Folgenden seien $\mathbf{v}_k := \mathbf{v}_0, \mathbf{v}_{k+1} := \mathbf{v}_1$. Es soll gelten, daß die Eckpunkte mathematisch positiv angeordnet sind, wenn man gegen den Normalenvektor auf die Fläche blickt:

$$\forall\, 0 \leq i < k:\ \det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{v}_{i+2} - \mathbf{v}_i] > 0$$

Die Determinante ist positiv genau dann, wenn die drei Spaltenvektoren in ihrer Reihenfolge ein Rechtssystem bilden. Das bedeutet anschaulich, daß die nächste Ecke links von der vorherigen Kantengeraden liegt; man macht sich das am besten anhand der Abbildung 3.1 mittels der Rechte-Hand-Regel klar.

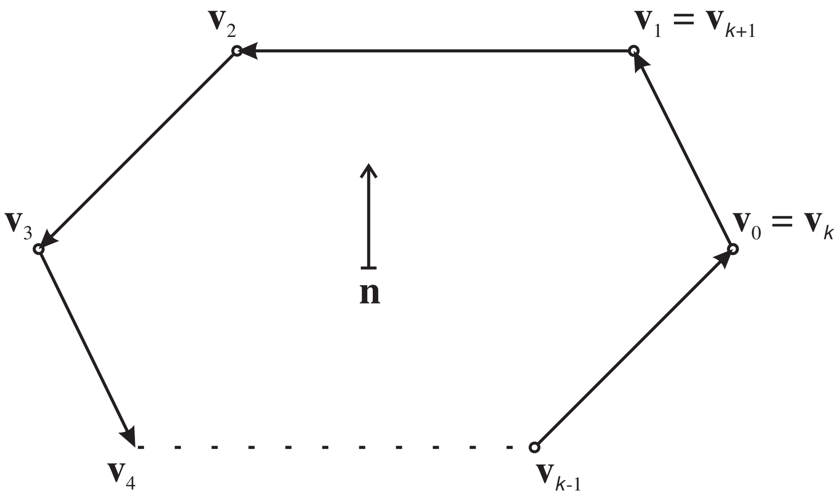


Abbildung 3.1: Schema eines Polygons

Die eigentliche Fläche ergibt sich damit als Punktmenge

$$\begin{aligned} f := f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}} &:= \{\sum_{i=0}^{k-1} \lambda_i \mathbf{v}_i \mid \sum_{i=0}^{k-1} \lambda_i = 1,\ \forall\, 0 \leq i < k:\ 0 \leq \lambda_i \in \mathbb{R}\} \\ &= \{x \in E_{\mathbf{n},n_0} \mid \forall\, 0 \leq i < k:\ \det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{x} - \mathbf{v}_i] \geq 0\} \subseteq \mathbb{R}^3 \end{aligned}$$

Die erste Form zeigt $f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}}$ als Menge der Konvexkombinationen der Ecken, die zweite beschreibt die Punkte von f als jene, welche links von allen Kanten liegen; dabei wird f wieder von außen betrachtet.

Wir unterscheiden nun drei Fälle nach der Lage der orientierten Geraden durch $l_{\mathbf{p},\mathbf{q}}$ zur Ebene $E_{\mathbf{n},n_0}$.

1. Fall: „nach außen“

   Es gilt $\mathbf{n}^{\mathrm{T}}(\mathbf{q}-\mathbf{p}) > 0$. Die Gerade schneidet die Ebene dann in genau einem Punkt $\mathbf{s}$. Dieser liegt auf dem Liniensegment genau dann, wenn dessen Anfang im inneren und dessen Ende im äußeren abgeschlossenen Halbraum der Ebene liegt. Daß der Schnittpunkt zur Fläche f gehört, kann man wie oben gezeigt daran erkennen, daß er links von allen Randkanten liegt. Damit ergibt sich für diesen Fall:

$$l_{\mathbf{p},\mathbf{q}} \cap f \neq \emptyset \iff$$
$$\mathbf{n}^{\mathrm{T}}\mathbf{q} \geq n_0 \;\wedge\; \mathbf{n}^{\mathrm{T}}\mathbf{p} \leq n_0 \;\wedge\; \forall\, 0 \leq i < k :\; \det[\mathbf{q}-\mathbf{p}, \mathbf{v}_{i+1}-\mathbf{v}_i, \mathbf{p}-\mathbf{v}_i] \geq 0$$

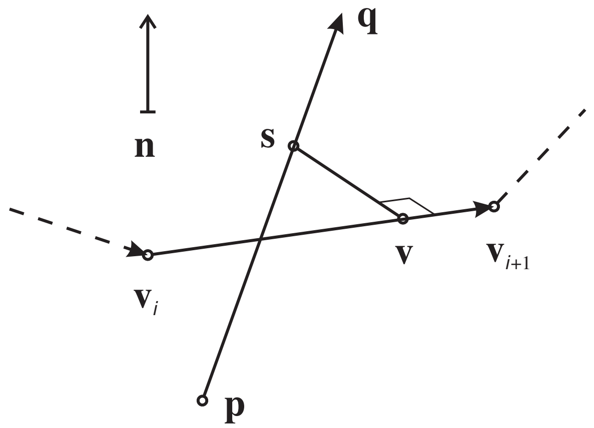


Abbildung 3.2: Schnitt zwischen Liniensegment und Fläche

Der Leser mag sich wundern, warum in dieser Formel der Schnittpunkt $\mathbf{s}$ nicht mehr auftritt. Es ist in der Tat so, daß der Schnittest ohne Verwendung von $\mathbf{s}$ formuliert werden kann. Dazu betrachtet man Abbildung 3.2 und zu einer Randkante $l_{\mathbf{v}_i,\mathbf{v}_{i+1}}$ den Lotfußpunkt $\mathbf{v}$ von $\mathbf{s}$ auf diese Kante. Der Punkt $\mathbf{s}$ liegt links von der Kantengeraden — wenn man die Außenseite von f betrachtet —, falls die Vektoren $\mathbf{n}$, $\mathbf{v}_{i+1}-\mathbf{v}_i$ und $\mathbf{s}-\mathbf{v}$ in dieser Reihenfolge ein Rechtssystem bilden. Letztere Bedingung drückt man durch eine Determinante aus, die nur Null wird, falls $\mathbf{s}=\mathbf{v}$, der Schnittpunkt also auf der Kantengeraden liegt. Nun nutzt man aus, daß $\mathbf{v}$ durch $\mathbf{v}_i$ und $\mathbf{v}_{i+1}$ darstellbar ist, ebenso $\mathbf{s}$ durch $\mathbf{p}$ und $\mathbf{q}$. Ferner sind für $\mathbf{s}\neq\mathbf{v}$ die drei Vektoren $\mathbf{q}-\mathbf{p}$, $\mathbf{v}_{i+1}-\mathbf{v}_i$ und $\mathbf{s}-\mathbf{v}$ linear unabhängig und ergeben daher durch eine geeignete Linearkombination die Flächennormale $\mathbf{n}$:

$$\mathbf{n}^{\mathrm{T}}(\mathbf{q}-\mathbf{p}) > 0 \;\wedge\; \mathbf{s}\neq\mathbf{v} \Rightarrow$$

$$\exists\, \alpha > 0, \beta, \gamma : \ \alpha(\mathbf{q} - \mathbf{p}) = \mathbf{n} + \beta(\mathbf{v}_{i+1} - \mathbf{v}_i) + \gamma(\mathbf{s} - \mathbf{v})$$

Dadurch werden folgende Umformungen verständlich:

$$\begin{array}{rl} & \det[\mathbf{v}_{i+1} - \mathbf{v}_i,\ \mathbf{s} - \mathbf{v},\ \mathbf{n}] \\ = & \det[\mathbf{v}_{i+1} - \mathbf{v}_i,\ \mathbf{s} - \mathbf{v},\ \mathbf{n} + \beta(\mathbf{v}_{i+1} - \mathbf{v}_i) + \gamma(\mathbf{s} - \mathbf{v})] \\ = & \det[\mathbf{v}_{i+1} - \mathbf{v}_i,\ \mathbf{s} - \mathbf{v},\ \alpha(\mathbf{q} - \mathbf{p})] \\ = & \alpha \det[\mathbf{v}_{i+1} - \mathbf{v}_i,\ \underbrace{\mathbf{s} + \mu(\mathbf{q} - \mathbf{p})}_{\mathbf{p}} - \underbrace{\mathbf{v} + \nu(\mathbf{v}_{i+1} - \mathbf{v}_i)}_{\mathbf{v}_i},\ \mathbf{q} - \mathbf{p}] \\ = & \alpha \det[\mathbf{v}_{i+1} - \mathbf{v}_i,\ \mathbf{p} - \mathbf{v}_i,\ \mathbf{q} - \mathbf{p}] \end{array}$$

woraus man

$$\det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{s} - \mathbf{v}] \geq 0 \iff \det[\mathbf{q} - \mathbf{p}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \geq 0$$

und damit obige Formel folgert. Diese Determinante läßt sich auch vierdimensional fassen, dann wird die Symmetrie der beteiligten Punkte deutlicher. Durch Entwicklung nach der ersten Zeile, Zusammenfassen und Umformen gemäß der Rechenregeln für Determinanten erkennt man

$$\begin{array}{rl} & \det \begin{pmatrix} 1 & 1 & 1 & 1 \\ \mathbf{p} & \mathbf{q} & \mathbf{v}_i & \mathbf{v}_{i+1} \end{pmatrix} \\ = & \det[\mathbf{q} - \mathbf{p}, \mathbf{v}_i, \mathbf{v}_{i+1}] + \det[\mathbf{p}, \mathbf{q}, \mathbf{v}_{i+1} - \mathbf{v}_i] \\ = & \det[\mathbf{q} - \mathbf{p}, \mathbf{v}_i - \mathbf{q}, \mathbf{v}_{i+1} - \mathbf{v}_i] \\ = & \det[\mathbf{q} - \mathbf{p}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \end{array}$$

2. Fall: „nach innen“

   Es gilt $\mathbf{n}^{\mathrm{T}}(\mathbf{q} - \mathbf{p}) < 0$. Dieser Fall ist symmetrisch zu obigem, d.h. durch Vertauschen von $\mathbf{p}$ und $\mathbf{q}$ ergibt sich:

$$l_{\mathbf{p},\mathbf{q}} \cap f \neq \emptyset \iff$$

$$\mathbf{n}^{\mathrm{T}}\mathbf{p} \geq n_0 \ \wedge\ \mathbf{n}^{\mathrm{T}}\mathbf{q} \leq n_0 \ \wedge\ \forall\, 0 \leq i < k : \ \det[\mathbf{p} - \mathbf{q}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{q} - \mathbf{v}_i] \geq 0$$

3. Fall: „parallel“

   Es gilt $\mathbf{n}^{\mathrm{T}}(\mathbf{q} - \mathbf{p}) = 0$. Falls $\mathbf{p} \notin E_{\mathbf{n},n_0}$ liegt kein Schnitt vor, daher sei ab jetzt $\mathbf{n}^{\mathrm{T}}\mathbf{p} = n_0$. Dies ist der undankbare der drei Fälle, da die beschreibende Formel recht kompliziert wird, aber letztlich nicht viel zur Kollisionserkennung beiträgt. Wichtig ist es, Überlappungen des offenen Inneren der Polyeder zu erkennen, denn dann liegt eine wirklich unfreie Position vor. Plazierungen der Polyeder, bei denen sich lediglich deren Oberflächen schneiden, werden dagegen als Kontakte bzw. halbfreie Positionen bezeichnet. Bei einer Bewegung sollte es erlaubt sein, sowohl freie als auch halbfreie Plazierungen zu benutzen; dies ist unabdingbar, wenn z.B.

kürzeste Wege gesucht sind. Aufgrund der Vorgehensweise der hier vorgestellten heuristischen Strategie, die insbesondere nicht vollständig ist, können halbfreie Positionen nach Belieben als frei oder unfrei angesehen werden. Dadurch gehen in der Praxis keine Lösungen verloren, und die Beschreibung des Algorithmus kann vereinfacht werden. Deshalb soll die Kollisionsrechnung nur garantieren, daß unfreie und freie Positionen als solche erkannt werden, halbfreie dürfen je nach Zweckmäßigkeit klassifiziert werden.

In diesem Sinne wird dieser Fall also in einer Implementierung ignoriert; dennoch werden die Formeln der Vollständigkeit halber genannt. Die Idee dabei ist ähnlich der beim Schnitt von Polyedern verwendeten. Ein Schnitt zwischen Kante und Fläche liegt genau dann vor, wenn ein Endpunkt der Kante im Flächeninneren liegt oder die Kante den Rand der Fläche schneidet. Liegt dabei $l_{\mathbf{p},\mathbf{q}}$ parallel zu einer Randkante, so liegt entweder kein Schnitt mit dieser vor oder die Information wird bereits an anderer Stelle berechnet, und zwar weil $\mathbf{p} \in f$ ist oder $l_{\mathbf{p},\mathbf{q}}$ eine Ecke enthält und damit eine benachbarte Randkarte nicht entartet schneidet. Demnach gilt:

$$\begin{aligned} l_{\mathbf{p},\mathbf{q}} \cap f \neq \emptyset \quad &\Longleftrightarrow \quad (\forall\, 0 \le i < k: \ \det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \ge 0) \\ &\qquad \vee \quad \big(\exists\, 0 \le i < k: \ l_{\mathbf{p},\mathbf{q}} \not\parallel l_{\mathbf{v}_{i+1}-\mathbf{v}_i} \ \wedge \ S(\mathbf{p}, \mathbf{q}, \mathbf{v}_i, \mathbf{v}_{i+1})\big) \end{aligned}$$

$$l_{\mathbf{p},\mathbf{q}} \not\parallel l_{\mathbf{v}_{i+1}-\mathbf{v}_i} \iff (\mathbf{q} - \mathbf{p}) \times (\mathbf{v}_{i+1} - \mathbf{v}_i) \neq \mathbf{0}$$

Das Prädikat S zeigt an, ob sich zwei Liniensegmente schneiden; dies ist z.B. genau dann der Fall, wenn die Gerade durch das eine Segment die Endpunkte des anderen voneinander trennt. In Formeln:

$$\begin{aligned} S(\mathbf{p}, \mathbf{q}, \mathbf{v}_i, \mathbf{v}_{i+1}) \ := \ & \det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \cdot \det[\mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{q} - \mathbf{v}_i] \le 0 \\ \wedge \ & \det[\mathbf{n}, \mathbf{q} - \mathbf{p}, \mathbf{v}_i - \mathbf{p}] \cdot \det[\mathbf{n}, \mathbf{q} - \mathbf{p}, \mathbf{v}_{i+1} - \mathbf{p}] \le 0 \end{aligned}$$

Insgesamt bedeutet das für den gesuchten Algorithmus zur Kollisionsrechnung, daß er den Test auf Schnitt der Polyederoberflächen auf folgendes Prädikat zurückführt:

$$\begin{aligned} S(l_{\mathbf{p},\mathbf{q}}, f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}}) \ := \ & \Big(\mathbf{n}^{\mathrm{T}}(\mathbf{q} - \mathbf{p}) > 0 \ \wedge \ \mathbf{n}^{\mathrm{T}}\mathbf{q} - n_0 \ge 0 \ \wedge \ \mathbf{n}^{\mathrm{T}}\mathbf{p} - n_0 \le 0 \\ & \ \wedge \bigwedge_{0 \le i < k} \det[\mathbf{q} - \mathbf{p}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \ge 0\Big) \\ \vee \ & \Big(\mathbf{n}^{\mathrm{T}}(\mathbf{p} - \mathbf{q}) < 0 \ \wedge \ \mathbf{n}^{\mathrm{T}}\mathbf{p} - n_0 \ge 0 \ \wedge \ \mathbf{n}^{\mathrm{T}}\mathbf{q} - n_0 \le 0 \\ & \ \wedge \bigwedge_{0 \le i < k} \det[\mathbf{p} - \mathbf{q}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{q} - \mathbf{v}_i] \ge 0\Big) \end{aligned}$$

Sind nun zwei Polyeder P und Q gegeben, so werde die Menge ihrer Kanten mit E_P und E_Q bezeichnet (von engl. *edge*), die Menge der Flächen mit F_P und F_Q. Das vollständige Prädikat für den Schnitt der Oberflächen lautet dann:

$$S(P, Q) := \Big(\bigvee_{\{\mathbf{p},\mathbf{q}\} \in E_P} \bigvee_{f \in F_Q} S(l_{\mathbf{p},\mathbf{q}}, f)\Big) \vee \Big(\bigvee_{\{\mathbf{p},\mathbf{q}\} \in E_Q} \bigvee_{f \in F_P} S(l_{\mathbf{p},\mathbf{q}}, f)\Big)$$

Die Größe von $S(l_{\mathbf{p},\mathbf{q}}, f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}})$ hängt nur ab von der Anzahl k der Ecken von f und ist bezüglich dieser linear. Für die Komplexität von $S(P,Q)$ ergibt sich mithin $\Theta(|P||Q|)$.

Enthaltungen

Damit fehlt zum Erkennen der Überlappung zweier Polyeder nur noch eine Abfrage, ob der eine ganz in dem anderen enthalten ist. Dann gilt:

$$P \cap Q \neq \emptyset \iff S(P,Q) \ \vee \ P \subset Q \ \vee \ Q \subset P$$

Da der Test auf $P \subset Q$ nur in diesem Zusammenhang verwendet wird, kann man auch eine Obermenge der Fälle erfassen, in denen $P \subset Q$, solange dabei stets $P \cap Q \neq \emptyset$ gilt. Auf diese Weise gelangt man zu der deutlich einfacheren Frage, ob irgendein Punkt von P, z.B. eine beliebige Ecke $\mathbf{v}$, in Q liegt. Das entscheidet man leicht durch Betrachten eines Strahls von $\mathbf{v}$ aus in eine zufällige Richtung. Trifft der Strahl Ecken oder Kanten von Q, so wird eine neue Richtung gewürfelt, sonst zählt man die geschnittenen Flächen; eine ungerade Anzahl bedeutet dabei, daß $\mathbf{v} \in Q \Rightarrow \{\mathbf{v}\} \subset P \cap Q \neq \emptyset$. Gilt umgekehrt $P \subset Q$, so folgt natürlich auch $\mathbf{v} \in Q$; d.h. diese Abfrage genügt den eben gestellten Ansprüchen. Mit leichten Modifikationen kann dieser Test auch dann noch verwendet werden, wenn P und damit $\mathbf{v}$ sich bewegt, seine erwartete Laufzeit bleibt auch dann bei $O(|Q|)$.

3.2.3 Dynamische Kollisionserkennung

Nachdem das Prädikat zur Erkennung überlappender Polyeder vollständig erklärt ist, muß nun noch geschildert werden, wie man es im Falle eines bewegten Polyeders effizient auswertet, um alle Kollisionszeitpunkte zu erhalten. Dazu sollte aber zuerst geklärt werden, was unter einer Bewegung zu verstehen ist.

In diesem Kapitel ist damit eine Translation gemeint, die das bewegte Polyeder aus seiner aktuellen Lage in Richtung von $\mathbf{r} \in \mathbb{R}^3$, $|\mathbf{r}| = 1$, vollführt. Die Länge der Verschiebung wird kontrolliert durch einen Parameter $\lambda_{Max} \in \mathbb{R}$. Damit ergibt sich die allgemeine Translation als Abbildung

$$T_{\mathbf{r}}^{\lambda} : \mathbb{R}^3 \to \mathbb{R}^3, \ \mathbf{x} \mapsto T_{\mathbf{r}}^{\lambda}(\mathbf{x}) := \mathbf{x}^{\lambda} := \mathbf{x} + \lambda\mathbf{r}$$

und eine Bewegung ist gegeben durch die stetige Zuordnung $\Lambda : [0,1] \to \mathbb{R}$, $t \mapsto \Lambda(t) := t\lambda_{Max}$. Das Polyeder P befindet sich demnach zum Zeitpunkt t in der Position $T_{\mathbf{r}}^{\Lambda(t)}(P)$, wobei $t = 0$ dem Anfang und $t = 1$ dem Ende der Bewegung entspricht.

Der eben geschilderte Test, ob ein Punkt des Polyeders P in Q liegt, muß nun an diese Bewegung angepaßt werden. Dazu wird angenommen, daß P sich bewegt und Q fest bleibt; dies kann bei Translationen wegen der relativen Natur der Bewegung immer vorausgesetzt werden. Sodann betrachtet man einen geeigneten Punkt $\mathbf{v}$ von P — dazu später mehr — und von dort einen Strahl in Richtung $\mathbf{r}$, d.h. den Weg von $\mathbf{v}$

bei der Verschiebung $T_{\mathbf{r}}^{\lambda}$. Die Menge $\{\lambda \mid T_{\mathbf{r}}^{\lambda}(\mathbf{v}) = \mathbf{v}^{\lambda} \in Q\}$ ermittelt sich wie folgt. Alle Flächen von Q werden auf Schnitt mit dem Strahl geprüft, die Schnittpunkte werden sortiert. Da der Punkt $\mathbf{v}$ im Unendlichen sicher nicht in Q liegt, ergibt die Folge der Schnitte des Strahls mit den Flächen eindeutig eine Abfolge von Intervallen, in denen sich $\mathbf{v}^{\lambda}$ innerhalb und außerhalb von Q befindet.

Auf diese Weise sind alle Kollisionszeitpunkte von $\mathbf{v}$ und Q errechnet, sofern keiner der Schnitte entartet war. Bei einer Entartung werden mehrere Flächen gleichzeitig getroffen und es ist nicht ohne weiteres klar, ob ein Wechsel von Innen und Außen vorliegt. Sollte also eine Kante oder gar Ecke von Q getroffen werden, so wird einfach ein anderer Punkt von P gewählt und der Vorgang wiederholt; in der Richtung des Strahls besteht keine Wahlfreiheit. Die Suche nach einem geeigneten $\mathbf{v}$ kann sicher auf die Oberfläche von P beschränkt werden, dabei sogar auf ein Dreieck, dessen Normalenvektor nicht senkrecht zu $\mathbf{r}$ steht; nicht alle Punkte eines solchen Dreiecks können zu entarteten Schnitten führen. Dies sieht man leicht ein, indem man die Projektion der Kanten von Q in eine Ebene senkrecht zu $\mathbf{r}$ betrachtet; nur Punkte von P, die auf solche Kanten projiziert werden, führen zu den erwähnten Entartungen.

Ein geeignetes Dreieck läßt sich offensichtlich in Zeit $O(|P|)$ finden, der Kollisionstest von $\mathbf{v}$ und Q erfolgt ebenfalls in Linearzeit $O(|Q|)$. Wählt man innere Punkte des Dreiecks zufällig und gleichverteilt aus, so trifft $\mathbf{v}^{\lambda}$ ein Kante von Q mit Wahrscheinlichkeit 0. Damit wird sich dieser Anteil des Kollisionstests mit $O(|P| + |Q|)$ als unbedeutend für die Gesamtlaufzeit erweisen.

Kommen wir nun zum Schnitt der Polyederoberflächen. Um die Auswertung von $S(l_{\mathbf{p},\mathbf{q}}, f)$ zu erleichtern, sei die Bewegung derart als Relativbewegung beschrieben, daß das Liniensegment verschoben wird und die Fläche fest bleibt. Für eine feste Translationsrichtung gilt es jetzt, alle relevanten Kollisionszeitpunkte zu bestimmen, d.h. die Menge

$$\begin{aligned} \mathcal{L} &:= \{\lambda \in \mathbb{R} \mid S(T_{\mathbf{r}}^{\lambda}(l_{\mathbf{p},\mathbf{q}}), f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}})\} \\ &\subseteq \{\lambda \in \mathbb{R} \mid T_{\mathbf{r}}^{\lambda}(l_{\mathbf{p},\mathbf{q}}) \cap f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}} \neq \emptyset\} \end{aligned}$$

Diese Menge kann dann noch eingeschränkt werden auf $[0, \lambda_{Max}]$, da nur diese Positionen durch die Bewegung durchlaufen werden.

Mit obiger Notation gilt $T_{\mathbf{r}}^{\lambda}(l_{\mathbf{p},\mathbf{q}}) = l_{\mathbf{p}^{\lambda},\mathbf{q}^{\lambda}}$, was anschaulich bedeutet, daß das verschobene Liniensegment durch die verschobenen Endpunkte bestimmt ist. Das Prädikat $S(l_{\mathbf{p},\mathbf{q}}, f)$ besteht nun aus zwei Formen von Ungleichungen. Dabei steht $\mathcal{R}$ stellvertretend für ein beliebiges Vergleichssymbol $<, \leq, =, \geq, >$. Mit $\mathcal{R}^{-}$ sei das Symbol gemeint, das durch Vertauschen von $<$ und $>$ aus $\mathcal{R}$ hervorgeht.

1. $\mathbf{n}^{\mathrm{T}}\mathbf{x}^{\lambda} - c \; \mathcal{R} \; 0$

   Formeln dieser Art geben Auskunft über die relative Richtung zweier Vektoren $(\mathbf{n}^{\mathrm{T}}(\mathbf{q} - \mathbf{p}) > 0)$ bzw. darüber, auf welcher Seite einer Ebene ein Punkt liegt

($\mathbf{n}^{\mathrm{T}}\mathbf{p} - n_0 \leq 0$). Während der Bewegung bleibt f und damit $\mathbf{n}$ fest, $\mathbf{p}$ und $\mathbf{q}$ ändern sich.

$$\begin{aligned} \mathbf{n}^{\mathrm{T}}(\mathbf{x} + \lambda\mathbf{r}) - c \;\mathcal{R}\; 0 &\iff \\ \lambda\,\underbrace{\mathbf{n}^{\mathrm{T}}\mathbf{r}}_{a} + \underbrace{\mathbf{n}^{\mathrm{T}}\mathbf{x} - c}_{b} \;\mathcal{R}\; 0 &\iff \left\{ \begin{array}{llll} \lambda & \mathcal{R} & -b/a & \text{falls} \quad a > 0 \\ 0 & \mathcal{R} & -b & \text{falls} \quad a = 0 \\ \lambda & \mathcal{R}^{-} & -b/a & \text{falls} \quad a < 0 \end{array} \right. \end{aligned}$$

Man erkennt, daß in jedem der drei Fälle maximal ein Intervall von zulässigen λ-Werten entsteht, dieses kann auch ganz $\mathbb{R}$ sein. Die Ungleichung ist eventuell unlösbar.

2. $\det[\mathbf{x}, \mathbf{w}, \mathbf{p}^{\lambda} - \mathbf{v}] \;\mathcal{R}\; 0$

   Wegen $\mathbf{q}^{\lambda} - \mathbf{p}^{\lambda} = \mathbf{q} + \lambda\mathbf{r} - (\mathbf{p} + \lambda\mathbf{r}) = \mathbf{q} - \mathbf{p}$ ist dies die allgemeine Form der im Prädikat verwendeten Abfrage $\det[\mathbf{q} - \mathbf{p}, \mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{p} - \mathbf{v}_i] \geq 0$, falls $\mathbf{p}$ und $\mathbf{q}$ bewegt werden.

   $$\det[\mathbf{x}, \mathbf{w}, \mathbf{p}^{\lambda} - \mathbf{v}] = (\mathbf{x} \times \mathbf{w})^{\mathrm{T}}(\mathbf{p} + \lambda\mathbf{r} - \mathbf{v}) = \lambda\,\underbrace{(\mathbf{x} \times \mathbf{w})^{\mathrm{T}}\mathbf{r}}_{a} + \underbrace{(\mathbf{x} \times \mathbf{w})^{\mathrm{T}}(\mathbf{p} - \mathbf{v})}_{b}$$

   führt demnach analog zum 1. Fall zu einer linearen Ungleichung, deren Lösungen wieder höchstens ein reelles Intervall bilden.

Wir haben also gesehen, daß das Prädikat S für den Schnitt von Kante und Fläche und damit das für den Schnitt von Polyederoberflächen aus solchen Ungleichungen als atomaren Bestandteilen aufgebaut ist, deren Lösungsmenge jeweils maximal ein Intervall darstellt. Die entscheidende Idee ist nun, bei der Auswertung des Prädikats nicht wie im stationären Fall nur *wahr* und *falsch* mit den Zwischenergebnissen zu assoziieren. Vielmehr wird als Teilresultat eine Liste von Intervallen verwaltet, die für die Atome die Lösungsmenge der Ungleichungen beschreibt. Die Booleschen Operationen $\wedge$ und $\vee$ werden sodann auf den Intervallisten als Schnitt bzw. Vereinigung realisiert. So führt eine Auswertung des Prädikats zur Menge $\mathcal{L}$ aller λ-Werte, für die es wahr ist.

Eine Baumdarstellung des logischen Ausdrucks resultiert dabei in einer Divide-And-Conquer-Strategie bei der Berechnung von $\mathcal{L}$. Die Anzahl der Blätter des Baums für $S(P, Q)$ ist $\Theta(n)$ wegen der Größe des Prädikats, $n := |P||Q|$; die Tiefe eines balancierten Baums kann daher auf $O(\log n)$ beschränkt werden. Die Lösungsintervalle für die atomaren Ungleichungen ergeben sich jeweils in konstanter Zeit, die Gesamtzahl der Intervalle ist maximal n. Durch Vereinigung von Intervallisten kann die Anzahl einzelner Intervalle höchstens verkleinert werden; dies gilt ebenso für den Schnitt, bei dem die Anzahl der Randpunkte und damit die Gesamtzahl der Intervalle nicht vergrößert wird. Insgesamt können die kollisionsbehafteten Abschnitte der Bewegung also in Zeit $O(n \log n)$ berechnet werden.

Zur Erläuterung des Konzepts wird nun noch ein Algorithmus angegeben, mit dem zwei sortierte Listen von disjunkten Intervallen in Linearzeit zu einer ebensolchen Ergebnisliste vereinigt werden können. Ähnlich dazu kann auch der Schnitt in Linearzeit berechnet werden, das Verfahren ist sogar etwas einfacher.

In der hier dargestellten Version (siehe Abbildung 3.3, Seite 47) werden alle Intervalle als abgeschlossen betrachtet; dadurch wird der Abschluß von $\mathcal{L}$ berechnet sowie eventuell einige isolierte Punkte, die als halbfreie Positionen für die Zwecke der Bewegungsplanung ignoriert werden dürfen. Der Übergang zu abgeschlossenen Konfigurationsraumhindernissen stört die hier betrachtete heuristische Bewegungsplanungsstrategie nicht; die Implementierung wird aber dadurch vereinfacht. Ferner sollte man bedenken, daß die Unterscheidung zwischen abgeschlossenen und offenen Intervallen mittels Fließkommaarithmetik unmöglich ist, dazu wirken sich Rundungsfehler auch in einfachen praktischen Beispielen schon zu stark aus.

Leere Intervallisten lassen sich repräsentieren als Folge der Länge 0; die Menge $\mathbb{R}$ kann ersetzt werden durch $[0, \lambda_{Max}]$, denn nur für diesen Bereich interessieren wir uns hier.

geg.:

$$l = [[a_1, z_1], \ldots, [a_m, z_m]], \quad \forall\, 1 \le i < m : \; z_i < a_{i+1}$$
$$l' = [[a'_1, z'_1], \ldots, [a'_n, z'_n]], \quad \forall\, 1 \le j < n : \; z'_i < a'_{j+1} \qquad \text{Intervallisten}$$

ges.:

$$\hat{l} = [[\hat{a}_1, \hat{z}_1], \ldots, [\hat{a}_k, \hat{z}_k]], \quad \forall\, 1 \le h < k : \; \hat{z}_h < \hat{a}_{h+1} \qquad \text{Ergebnisliste}$$
$$\text{mit } \bigcup_{h=1}^{k} [\hat{a}_h, \hat{z}_h] = \bigcup_{i=1}^{m} [a_i, z_i] \;\cup\; \bigcup_{j=1}^{n} [a'_j, z'_j]$$

Aufruf:

$$\hat{l} \leftarrow \text{unite}\,(l, l')$$

3.3 Die translatorische Strategie

Mit der Kollisionsrechnung steht nun das nötige Hilfsmittel zur Verfügung, um die heuristische Strategie zur Bewegungsplanung anzuwenden auf einen Körper, der bei konstanter Orientierung frei im Raum verschiebbar ist. Wie bereits angekündigt wird das Problem übersetzt in die Bewegung eines Punktes im Konfigurationsraum des Körpers. Für die weiteren Erläuterungen gehen wir von folgender Problemstellung aus:

geg.:

$P, Q_1, \ldots, Q_n$: Polyeder — bewegtes Objekt und Hindernisse

$\mathbf{v} \in \mathbb{R}^3$ — Verschiebungsvektor

```
unite (l, l' : IntervalList) : IntervalList
    weiter : Boolean
    i, j : ℕ
    a, z : ℝ
    l̂ : IntervalList

l̂ ← []                                          anfangs leere Liste
i ← j ← 1                                        aktuelle Intervalle
while i ≤ m ∧ j ≤ n do                           solange noch keine Liste abgearbeitet
    if z_i < a'_j then
        l̂ ← l̂ ∘ [a_i, z_i]                        [a_i, z_i] vor [a'_j, z'_j]
        i ← i + 1
    else if z'_j < a_i then
        l̂ ← l̂ ∘ [a'_j, z'_j]                      [a'_j, z'_j] vor [a_i, z_i]
        j ← j + 1
    else
        a ← min(a_i, a'_j)                       Intervalle überlappen,
        z ← max(z_i, z'_j)                       bilde Vereinigung [a, z]
        i ← i + 1
        j ← j + 1                                betrachte nachfolgende Intervalle
        weiter ← true
        while weiter do                          Ende der Überlappung suchen
            if i ≤ m ∧ a_i ≤ z then
                z ← max(z, z_i)                  [a_i, z_i] überlappt [a, z]
                i ← i + 1
            else if j ≤ n ∧ a'_j ≤ z then
                z ← max(z, z'_j)                 [a'_j, z'_j] überlappt [a, z]
                j ← j + 1
            else
                weiter ← false                   Ende der Überlappung erreicht
        l̂ ← l̂ ∘ [a, z]                            Vereinigung [a, z] ans Ergebnis anhängen

if i < m then
    l̂ ← l̂ ∘ [[a_i, z_i], ..., [a_m, z_m]]         Rest von Liste l übrig
else if j < n then
    l̂ ← l̂ ∘ [[a'_j, z'_j], ..., [a'_n, z'_n]]     Rest von Liste l' übrig
return l̂
```

Abbildung 3.3: Vereinigung zweier Intervallisten

ges.:

$pfad : \text{List}(\mathbb{R}^3)$ berechneter Weg

Aufruf:

$pfad \leftarrow \text{connect}(P, Q_1, \ldots, Q_n, \mathbf{0}, \mathbf{v}, [\,])$ [ ] sei leere Liste

Sowohl der zu bewegende Körper als auch die festen Hindernisse seien durch Polyeder modelliert, und zwar in ihrer jeweiligen Ausgangslage. Ferner ist eine Ziellage beschrieben durch den Translationsvektor $\mathbf{v}$, d.h. das Objekt P soll aus der Anfangsposition $\mathbf{0}$ in die Endposition $\mathbf{v}$ gebracht werden. Als Ergebnis der Planung wird ein Weg für P erwartet, der bzgl. der Hindernisse Q_1 bis Q_n kollisionsfrei ist.

Nach einem erfolgreichen Aufruf enthält die Liste $pfad$ die Beschreibung eines solchen Weges, der stückweise linear verläuft. Scheitert die Heuristik, so gibt sie eine leere Liste zurück. Sei also $pfad = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, $0 < k \in \mathbb{N}$, und $\mathbf{v}_0 := \mathbf{0}$. Dann gilt $\mathbf{v}_k = \mathbf{v}$, d.h. der Weg führt zum Ziel, und die folgende Abbildung

$$\tau : [0,1] \to \mathbb{R}^3,\ t \mapsto \begin{cases} \mathbf{v}_i & kt = i \\ \mathbf{v}_i + (kt - i)(\mathbf{v}_{i+1} - \mathbf{v}_i) & i < kt < i+1 \end{cases}$$

ist stetig mit $\tau(0) = \mathbf{0}$ und $\tau(1) = \mathbf{v}$, d.h. sie stellt einen Weg von $\mathbf{0}$ nach $\mathbf{v}$ dar. Dieser ist kollisionsfrei in dem Sinne, daß

$$\forall\, 0 \le t \le 1\ \forall\, 1 \le i \le n:\ T_{\tau(t)}(P) \cap Q_i = \emptyset$$

und liefert damit die gesuchte kollisionsfreie Bewegung für P.

Wir wissen also, wie der Bewegungsplanungsalgorithmus aufzurufen und wie sein Ergebnis zu deuten ist. Worin aber unterscheidet sich die interne Arbeitsweise der translatorischen Strategie für einen Körper von der bereits bekannten für den Fall eines Punktes im $\mathbb{R}^3$? Die Planungsheuristik arbeitet genau so, als wolle sie einen Weg für einen Punkt von $\mathbf{0}$ nach $\mathbf{v}$ finden. Die wesentliche Neuerung ist, daß keine direkte Repräsentation der Hindernisse für den Punkt, also der Konfigurationsraumhindernisse, bekannt ist. Vielmehr ergeben sich Informationen über diese stets implizit aus den Objektraumhindernissen und dem bewegten Körper mittels der Kollisionsrechnung. Eine explizite Darstellung, das sei nochmals betont, wird zu keinem Zeitpunkt errechnet. Da die Strategie zur Wegesuche jedoch nur eindimensionales Wissen über den Konfigurationsraum verwendet, ist dies auch nicht nötig. Ansonsten, insbesondere was Rekursions- und Ausweichstrategien betrifft, gilt das in vorangegangenen Kapiteln Gesagte.

3.4 Anmerkungen zur Implementierung

Die geschilderte Auswertung eines Prädikates wurde nicht in einer logischen Programmiersprache realisiert, sondern in einer imperativen, genauer gesagt in C++. Dabei können einige spezielle Eigenschaften des Prädikats ausgenutzt und heuristische Ansätze zur Beschleunigung eingesetzt werden.

Zum Beispiel bemerkt man, daß im Prädikat $S(l_{\mathbf{p},\mathbf{q}}, f)$ eine Fallunterscheidung nach der Richtung der Linie relativ zum Normalenvektor vorgenommen wird. Durch Vertauschen der Endpunkte $\mathbf{p}$ und $\mathbf{q}$ kann man eine Normalform des Problems erreichen, d.h. es muß nur ein Fall wirklich ausführlich programmiert werden.

Sodann kann man die Auswertung der Konjunktion abbrechen, sobald eine leere Intervalliste entstanden ist. Das ist insbesondere dann der Fall, wenn das Liniensegment während der Bewegung auf einer Seite der Flächenebene bleibt. Ein einfacher Test der Endpunkte zu Beginn und am Ende der Verschiebung gibt hierüber Auskunft.

Es bieten sich also einige mehr oder weniger triviale Verbesserungsmöglichkeiten. Wirklich entscheidend für die praktische Laufzeit ist nun aber eine Technik, die möglichst viele Paare von Objekten rasch erkennt, welche bei der gegebenen Translation nicht kollidieren können. Solche Objekte können sowohl die gesamten Körper als auch deren Flächen und Kanten sein. Die zentrale Idee ist dabei, leicht zu handhabende Hüllen statt der eigentlichen Objekte zu testen; kollidieren diese nicht, so tun es auch die eingeschlossenen Punktmengen nicht. Man muß allerdings abwägen zwischen dem Aufwand, eine „gut passende" Hülle zu errechnen und in Tests zu benutzen, und der schlechten Aussagefähigkeit zu großer Hüllen.

Das IGOR -System zur **I**nteraktiven **G**raphischen **O**bjektmontage und **R**obotersimulation inklusive Kollisionserkennung stellt die Basis dar, welche im Rahmen dieser Diplomarbeit um ein Modul zur Bewegungsplanung erweitert wurde. Datenstrukturen zur Repräsentation von Polyedern und viele Algorithmen zum Umgang damit waren bereits vorhanden, auch solche zur Berechnung von Hüllkörpern. Für alle beteiligten Körper werden einmalig kleinste umhüllende Kugeln berechnet und mit dem Körper assoziiert, so daß sie jederzeit zur Verfügung stehen. Das implementierte Verfahren berechnet die Kugel mittels des Downhill Simplex Algorithmus als Minimum der Radiusfunktion für einen Mittelpunkt. Zum Zwecke der beschleunigten Erkennung kollisionsfreier Situationen ist es aber eigentlich nur wichtig, schnell einen kleinen Hüllkörper zu berechnen, auch wenn er den kleinsten nur annähert.

Damit sieht die verwendete Hierarchie bei der Benutzung von Hüllkörpern so aus, daß zuerst die Kugeln der Körper gegeneinander getestet werden. Ob diese bei der Translation disjunkt bleiben, kann mit einer einfachen quadratischen Ungleichung entschieden werden. Seien also $\mathbf{p}_i \in \mathbb{R}^3$, $0 < r_i \in \mathbb{R}$, $i = 1, 2$, die Mittelpunkte bzw. Radien der beiden Hüllkugeln, ferner sei eine Verschiebungsrichtung $\mathbf{r} \in \mathbb{R}^3$ gegeben. Bei einer Translation des ersten Körpers um $\mathbf{r}$ kommt es zur Kollision der Kugeln, falls

$$\begin{array}{rrcl} & |\mathbf{p}_1 + \lambda\mathbf{r} - \mathbf{p}_2| & \leq & r_1 + r_2 \\ \Longleftrightarrow & \lambda^2|\mathbf{r}|^2 + 2\lambda\mathbf{r}^{\mathrm{T}}(\mathbf{p}_1 - \mathbf{p}_2) + |\mathbf{p}_1 - \mathbf{p}_2|^2 & \leq & |r_1 + r_2|^2 \end{array}$$

für ein $\lambda \in [0, 1]$.

Falls die Kugeln sich während der Bewegung schneiden, werden die eingeschlossenen Körper genauer betrachtet. Dies geschieht in einem speziellen orthogonalen Koordinatensystem, dessen z-Achse gleich der Translationsrichtung ist. Damit beschreiben die x- und y-Koordinaten die Projektion des Körpers in eine Ebene senkrecht zum Verschiebungsvektor $\mathbf{r}$ und bleiben während der Translation gleich. Als Hüllkörper werden sogenannte Isoboxen berechnet, also achsenparallele umschließende Quader, und zwar für den gesamten Körper P und jede einzelne Fläche. Dies geschieht durch einfache Minimum-Maximum-Bildung über die Koordinaten der Ecken; damit ergeben sich diese Isoboxen als kartesisches Produkt von x-, y- und z-Intervallen. Eine solche Berechnung erfolgt in Linearzeit bzgl. $|P|$; man merkt sich das Ergebnis, um es später wiederzuverwenden, wenn z.B. das bewegte Objekt gegen ein anderes Hindernis auf Kollision getestet wird.

Nun ist klar, daß der bewegte Körper das Hindernis nicht berühren kann, falls die „Schatten" bei der Projektion sich nicht schneiden oder falls es eine Ebene senkrecht zu $\mathbf{r}$ gibt, die beide Objekte während der ganzen Bewegung trennt. Das ist sicherlich der Fall, wenn mindestens eines der Intervalle, welche die Isobox des einen Körpers beschreiben, disjunkt zu dem entsprechenden des anderen Körpers ist. Das z-Intervall des bewegten Körpers wird dabei so vergrößert, daß dieser während der Translation die Isobox nicht verläßt. Eine schöne Darstellung des Tests auf Kollisionsfreiheit zweier statischer Isoboxen ergibt sich, wenn die Intervalle jeder Koordinate als eindimensionale Kugeln dargestellt werden, wie es im IGOR -System daher auch geschieht. Seien dazu $p_i^\kappa \in \mathbb{R},\ 0 \le r_i^\kappa \in \mathbb{R},\ i \in \{1,2\},\ \kappa \in \{x,y,z\}$, die Mittelpunkte und Radien dieser Kugeln für beide Körper, getrennt nach den drei Koordinaten; der Hüllquader des ersten Körpers läßt sich dann beschreiben als

$$[p_1^x - r_1^x,\ p_1^x + r_1^x] \times [p_1^y - r_1^y,\ p_1^y + r_1^y] \times [p_1^z - r_1^z,\ p_1^z + r_1^z]$$

und die beiden Isoboxen überlappen sich genau dann nicht, wenn

$$|p_1^x - p_2^x| > r_1^x + r_2^x \ \vee\ |p_1^y - p_2^y| > r_1^y + r_2^y \ \vee\ |p_1^z - p_2^z| > r_1^z + r_2^z$$

Wird zwischen den Isoboxen von P und Q eine Überlappung festgestellt, so müssen zur Auswertung von $S(P,Q)$ letztlich sämtliche Kante-Fläche-Paare betrachtet werden. Dazu werden aber zuerst Fläche-Fläche-Paare betrachtet, und nur falls deren Hüllquader kollidieren, wird wirklich mittels der Randkanten der Flächen $S(l,f)$ ausgewertet. Dabei werden die eingangs erwähnten Verbesserungen im Programmablauf vorgenommen.

Für die weniger zeitaufwendige Abfrage, ob ein Körper bei der Verschiebung vollständig im anderen verschwindet, lassen sich die Isoboxen ebenfalls einsetzen. Wegen der Achsenparallelität der Hüllquader kann ein Objekt nur ganz in einem anderen liegen, wenn das auch für deren Hüllen gilt. Ferner muß beim Schnitt eines Strahls mit einer Fläche des Polyeders diese nicht genau betrachtet werden, wenn der Strahl die Isobox der Fläche nicht schneidet. Letzteres ist besonders einfach zu prüfen, da als Strahlrichtung die Translationsrichtung gewählt wird.

All diese erwähnten Techniken können an der asymptotischen worst-case-Komplexität der Probleme nichts ändern, sie verschwenden in ungünstigen Fällen sogar Zeit. Letztlich zeigt sich in praktischen Beispielen, insbesondere solchen, die auch Menschen als nicht zu schwer empfinden, jedoch deutlich, daß nur sehr wenige Hindernisse bei einer gegebenen Translation den Weg des Objekts stören. Daher ist es äußerst wichtig, den „unnützen" Hindernissen möglichst wenig Zeit zu widmen.

Kapitel 4

Erweiterung auf Rotationen

Dem zu bewegenden Objekt soll nun auch gestattet werden, seine Orientierung im Raum zu ändern. Das führt zu der Frage, wie die neu gewonnenen Freiheitsgrade zu nutzen sind. Aus Anwendersicht besteht die konzeptuell einfachste Lösung sicher darin, ohne Einschränkungen stetige Änderungen von Position und Orientierung zu erlauben. Dies führt jedoch zu einigen Schwierigkeiten, die es aus Sicht des Algorithmus einfacher erscheinen lassen, die Freiheitsgrade nach Translation und Rotation getrennt zu behandeln.

Historisch bedingt ist diese Trennung durch die Einbettung der Bewegungsplanung ins $\mathbb{ICOR}$-System. Dort werden die beiden Bewegungsarten separat gehandhabt, um effizientere Kollisionserkennung zu ermöglichen, insbesondere durch die in der Praxis sehr wichtigen „Faceboxing“-Tests. Da sich überlagerte Bewegungen auch beliebig exakt annähern lassen durch eine Folge reiner Rotationen und Translationen, ist der Verlust an Allgemeinheit zu verschmerzen, soll heißen, es können nach wie vor alle notwendigen Bewegungen zur Montage von Objekten simuliert werden. Damit nun automatisch geplante Bewegungen reibungslos ins Gesamtkonzept übernommen werden können, sollten diese von vorne herein die vorhandenen Beschränkungen berücksichtigen. So geplante Wege sind vom System sowohl leichter darzustellen als auch schneller hinsichtlich Kollisionsfreiheit zu verifizieren als solche, die durch Approximation aus einer überlagerten Bewegung hervorgehen.

Ferner werden die geplanten Wege nur durch Anfangs- und Endlage spezifiziert, und im Verlauf der Planungsstrategie ist auch der Begriff einer „direkten“ Verbindung zweier solcher Lagen notwendig. Die natürlichste Überlagerung der durch Anfangs- und Endlage gegebenen Verschiebung und Drehung ist nun eine, bei der beide Bewegungen mit konstanter Geschwindigkeit ablaufen. Dies führte bei der Implementierung aber zu Gleichungen, die nicht mehr geschlossen und exakt, sondern nur noch iterativ und näherungsweise zu lösen wären; das ginge sowohl zu Lasten der Laufzeit oder Genauigkeit als auch der Eleganz in der Realisierung. Einfach zu handhabende Wege entsprechen dagegen unnatürlich erscheinenden beschleunigten Bewegungen.

Drittens gibt es gewisse fundamentale Unterschiede zwischen den Parametern, die die Lage eines Punktes beschreiben, und denen, die die Orientierung eines Körpers wiederspiegeln. Letztere sind in gewissem Sinne zyklisch und müssen daher anders behandelt werden als erstere; das legt ebenfalls nahe, die beiden Arten von Freiheitsgraden zu trennen. Auf diese Weise gelangt man auch zu einer schönen Beschreibung der rotatorischen Strategie mit Hilfe der Kugelgeometrie.

Wie trennt man nun in der Planungsheuristik die beiden Arten von Freiheitsgraden? Dies geht analog zu Abschnitt 2.2, Seite 31, indem man die „direkte“ Verbindung zweier Lagen als Folge einer Rotation und Translation erklärt. Tritt auf einem der Teilstücke eine Kollision auf, so wird dieser ausgewichen; als Umwegpunkte kommen dabei eingeschränkte Bewegungen der gleichen Art oder beliebige der jeweils anderen Art in Frage. Zu beachten ist noch die Reihenfolge der Bewegungsarten auf dem Weg zum Umwegpunkt und von dort zum Ziel; sie hängt davon ab, durch welche Ausweichbewegung der Umwegpunkt entstand.

4.1 Parametrisierung der Lage eines Polyeders

In diesem Abschnitt geht es darum, wie die Lage eines im Raum frei beweglichen starren Körpers festgelegt werden kann, und durch welche Parameter man eine solche Lage geeignet beschreibt.

Dazu werde der dreidimensionale euklidische Anschauungsraum als affiner Raum über dem Vektorraum $\mathbb{R}^3$ mit Standardbasis aufgefaßt. Zwei beliebige Lagen des starren Körpers lassen sich dann durch eine affine Abbildung ineinander überführen; dieser wiederum läßt sich eindeutig eine Transformationsmatrix $\mathbf{U}$ und ein Verschiebungsvektor $\mathbf{v}$ zuordnen. Damit schreibt sich die affine Abbildung bzgl. $\mathbb{R}^3$ wie folgt: $\mathbf{x} \mapsto \mathbf{Ux}+\mathbf{v}$.

Sei umgekehrt $P \subseteq \mathbb{R}^3$ die Punktmenge, welche den Körper in einer beliebig gewählten Ausgangslage beschreibt; dann ergeben sich sämtliche möglichen Positionen des Körpers durch Anwendung einer Rotation und einer Translation.

Für $\mathbf{v} \in \mathbb{R}^3$ sei die Translation um $\mathbf{v}$ definiert als Abbildung

$$T_{\mathbf{v}} : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{x} \mapsto T_{\mathbf{v}}(\mathbf{x}) := \mathbf{x} + \mathbf{v}$$

Für eine unitäre Matrix $\mathbf{U} \in \mathbb{R}^{3\times 3}$, $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{E}$, $\det(\mathbf{U}) = 1$ sei die Rotation um $\mathbf{U}$ definiert als Abbildung

$$R_{\mathbf{U}} : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{x} \mapsto R_{\mathbf{U}}(\mathbf{x}) := \mathbf{Ux}$$

Unter dem Körper P in Lage $(\mathbf{v},\ \mathbf{U})$ verstehe man nun $P_{\mathbf{v},\mathbf{U}} := (T_{\mathbf{v}} \circ R_{\mathbf{U}})(P)$, den zuerst gedrehten und dann verschobenen Körper.

Die Verschiebung ist so bereits kompakt beschrieben, d.h. durch eine minimale Anzahl reeller Parameter, und dabei dennoch leicht zu handhaben. Für einen frei im Raum beweglichen Körper ist die Menge aller möglichen Verschiebungen gleich $\mathbb{R}^3$. Wir haben bereits gesehen, wie die heuristische Vorgehensweise Hindernissen in einem solchen Euklidischen Raum auszuweichen versucht.

Um eine ähnlich anschauliche geometrische Beschreibung der Strategie auch im rein rotatorischen Fall zu erhalten, bietet es sich an, Drehungen anders darzustellen als durch Matrizen. Jede Rotation $R_{\mathbf{U}}$ wie oben läßt sich beschreiben als Drehung um eine Achse $\mathbf{r} \in \mathbb{R}^3$, $|\mathbf{r}| = 1$, die durch den Ursprung verläuft, und einen bestimmten Winkel $\varphi \in \mathbb{R}$. Dem entsprechend gilt für die Matrix

$$\mathbf{U} = \mathbf{U}_{\mathbf{r},\varphi} = (1 - \cos\varphi)\mathbf{r}\mathbf{r}^{\mathrm{T}} + \cos\varphi \mathbf{E} + \sin\varphi \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix}$$

und umgekehrt für Winkel und Achse

$$\varphi = \arccos\left(\frac{1}{2}(u_{1,1} + u_{2,2} + u_{3,3} - 1)\right)$$

$$\mathbf{r} = (2\sin\varphi)^{-1}(u_{3,2} - u_{2,3}, u_{1,3} - u_{3,1}, u_{2,1} - u_{1,2})^{\mathrm{T}}$$

Das erlaubt uns, in Zukunft auch $R_{\mathbf{r},\varphi}$ für $R_{\mathbf{U}_{\mathbf{r},\varphi}}$ zu schreiben, die Rotation also direkt unter Angabe von Achse und Winkel zu benennen.

Die Anwendung einer so beschriebenen Rotation $R_{\mathbf{r},\varphi}$ auf einen Vektor $\mathbf{x} \in \mathbb{R}^3$ ergibt die Zuordnungsvorschrift

$$\mathbf{x} \mapsto (1 - \cos\varphi)\mathbf{r}^{\mathrm{T}}\mathbf{x}\mathbf{r} + \cos\varphi\mathbf{x} + \sin\varphi\mathbf{r} \times \mathbf{x} \tag{4.1}$$

die später noch nützlich sein wird. Man erkennt, daß die Drehung in einem Orthonormalsystem mit Basisvektoren $\mathbf{r}$, $\mathbf{x}$ und $\mathbf{r} \times \mathbf{x}$ ausgerechnet wird.

Eine elegante Darstellungsmöglichkeit für Rotationen und die rotatorische Strategie eröffnet sich durch Verwendung der Hamiltonschen *Quaternionen*, denen daher ein kurzer Abschnitt gewidmet wird.

4.1.1 Quaternionen

Die Menge $\mathbb{R}^4$ bildet mit der üblichen Vektoraddition und einer geeignet gewählten Multiplikation einen nichtkommutativen Ring mit Eins, in dem alle Elemente außer der Null invertierbar sind. Diese algebraische Struktur heißt auch Schiefkörper. Quaternionen $\mathbf{Q} = (q_0, q_1, q_2, q_3)^{\mathrm{T}} \in \mathbb{R}^4$ werden von nun an als $\mathbf{Q} = (q_0, \mathbf{q})$ geschrieben, q_0 wird auch Skalaranteil, $\mathbf{q} = (q_1, q_2, q_3)^{\mathrm{T}}$ Vektoranteil des Quaternions $\mathbf{Q}$ genannt. Dies ergibt in natürlicher Weise Inklusionen von $\mathbb{R}$ und $\mathbb{R}^3$ in die Menge $\mathbb{R}^4$ der Quaternionen. Als Multiplikation definiert man die bilineare Zuordnung

$$(p_0, \mathbf{p}) \cdot (q_0, \mathbf{q}) := (p_0 q_0 - \mathbf{p}^{\mathrm{T}}\mathbf{q},\ p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q})$$

Analog zu den komplexen Zahlen wird ein Quaternion $\mathbf{Q}$ *konjugiert* durch $\mathbf{Q} = (q_0, \mathbf{q}) \mapsto \mathbf{Q}^* := (q_0, -\mathbf{q})$, dabei gilt auch $|\mathbf{Q}|^2 = \mathbf{Q} \cdot \mathbf{Q}^*$. Ebenso ist $\mathbf{Q}^{-1} = \mathbf{Q}^*/|\mathbf{Q}|^2$.

Einer Rotation um die Achse $\mathbf{r}$ und den Winkel φ wird nun das Quaternion

$$\mathbf{Q}_{\mathbf{r},\varphi} = \left(\cos\frac{\varphi}{2}, \sin\frac{\varphi}{2}\mathbf{r}\right)$$

zugeordnet, wodurch die Abbildung $R_{\mathbf{r},\varphi}$ sich darstellt als

$$R_{\mathbf{r},\varphi} : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{x} \mapsto \mathbf{Q} \cdot (0, \mathbf{x}) \cdot \mathbf{Q}^*$$

mit Hilfe der „Einbettung“ von $\mathbb{R}^3$ in die Quaternionen und deren Multiplikation. An der Abbildungsvorschrift läßt sich auch erkennen, daß $\mathbf{Q}$ und $-\mathbf{Q}$ die gleiche Rotation repräsentieren; die Zuordnung von Quaternionen zu Matrizen ist daher nicht injektiv und umgekehrt kann man $\mathbf{r}$ und φ sowohl $\mathbf{Q}_{\mathbf{r},\varphi}$ als auch $\mathbf{Q}_{\mathbf{r},\varphi+2\pi}$ zuordnen.

Die Rotationsquaternionen $\mathbf{Q}_{\mathbf{r},\varphi}$ sind alle unitär, d.h. vom Betrag 1, und diese bilden eine Gruppe bzgl. des Quaternionenprodukts. Umgekehrt stellt jedes unitäre Quaternion $\mathbf{Q} = (q_0, \mathbf{q})$ eine Drehung dar, und zwar um den Winkel $\varphi = 2 \arccos q_0$; die Achse $\mathbf{r} = (\sin \frac{\varphi}{2})^{-1}\mathbf{q}$ ist nur für $\varphi \neq 0$ eindeutig bestimmt.

Damit können wir nun eine Rotation um $\mathbf{r}$ und φ auch durch ein Quaternion bezeichnen, d.h. $R_{\mathbf{r},\varphi} =: R_{\mathbf{Q}_{\mathbf{r},\varphi}}$. Die Hintereinanderausführung von Rotationen entspricht der Multiplikation der Quaternionen: $R_\mathbf{P} \circ R_\mathbf{Q} = R_{\mathbf{P}\cdot\mathbf{Q}}$.

4.1.2 Sphärische Geometrie

Sei $\mathcal{K}^n := \{\mathbf{x} \in \mathbb{R}^n \mid |\mathbf{x}| = 1\}$ die *Einheitskugelfläche* im $\mathbb{R}^n$. Speziell für $n = 4$ ergeben sich so die unitären Quaternionen als $\mathcal{K}^4$. Analog zu dort fassen wir $1 \in \mathbb{R}$ auf als $(1, 0, 0, 0)^\mathrm{T} \in \mathcal{K}^4$ und $\mathbf{r} \in \mathcal{K}^3$ als unitäres Quaternion $(0, \mathbf{r}) \in \mathcal{K}^4$.

Ein *Großkreis* auf $\mathcal{K}^n$, also ein Kreis um den Ursprung mit Radius 1, wird durch zwei zueinander senkrechte Achsen $\mathbf{P}, \mathbf{Q} \in \mathcal{K}^n$, $\mathbf{P} \perp \mathbf{Q}$ (bzgl. des Standardskalarprodukts im $\mathbb{R}^n$), festgelegt als

$$\{\mu\mathbf{P} + \nu\mathbf{Q} \mid \mu, \nu \in \mathbb{R};\ \mu^2 + \nu^2 = 1\} = \{\cos\varphi\mathbf{P} + \sin\varphi\mathbf{Q} \mid \varphi \in \mathbb{R}\}$$

wobei letztere Darstellung etwas anschaulicher ist. Ein solcher Großkreis wird eindeutig bestimmt durch die Ebene, in der er liegt, also durch zwei linear unabhängige Vektoren. Das führt zu

Lemma 4.1:
Durch zwei beliebige Punkte $\mathbf{P}, \mathbf{Q} \in \mathcal{K}^n, \mathbf{Q} \neq \pm\mathbf{P}$, *wird genau ein Großkreis festgelegt; als Achsen können* $\mathbf{P}$ *und* $\mathbf{R}^0$ *dienen, wobei* $\mathbf{R} := \mathbf{Q} - \mathbf{P}^\mathrm{T}\mathbf{Q}\ \mathbf{P}$ *und* $\mathbf{R}^0 := \mathbf{R}/|\mathbf{R}|$.

Beweis:
Wegen $\mathbf{Q} \neq \pm\mathbf{P}$ sind $\mathbf{P}$ und $\mathbf{R}$ linear unabhängig und spannen dieselbe Ebene auf wie $\mathbf{P}$ und $\mathbf{Q}$; zudem ist dann $\mathbf{R}$ nicht der Nullvektor und steht senkrecht auf $\mathbf{P}$:
$\mathbf{R} = 0 \iff \mathbf{Q} = \underbrace{\mathbf{P}^\mathrm{T}\mathbf{Q}}_{\lambda}\mathbf{P} \quad \Rightarrow |\lambda| = |\mathbf{Q}|/|\mathbf{P}| = 1 \quad \Rightarrow \mathbf{Q} = \pm\mathbf{P}$

$$\mathbf{R}^\mathrm{T}\mathbf{P} = \mathbf{Q}^\mathrm{T}\mathbf{P} - \mathbf{P}^\mathrm{T}\mathbf{Q}\underbrace{\mathbf{P}^\mathrm{T}\mathbf{P}}_{1} = 0 \quad \Rightarrow \mathbf{R} \perp \mathbf{P}$$

Für $\mathbf{R} \neq 0$ hat $\mathbf{R}^0$ die gleiche Richtung wie $\mathbf{R}$ und Länge 1; es ist $|\mathbf{R}| = \sqrt{1-\lambda^2}$. Demnach erzeugen die Achsen $\mathbf{P}$ und $\mathbf{R}^0$ einen Großkreis, der natürlich $\mathbf{P}$ und auch $\mathbf{Q} = \lambda\mathbf{P} + \sqrt{1-\lambda^2}\mathbf{R}^0$ enthält. ■

Zwischen Großkreisen und Rotationen besteht nun ein einfacher Zusammenhang. Stellen wir die Orientierung eines Körpers als Quaternion dar, so durchläuft diese während einer Volldrehung einen halben Großkreis, bzw. wegen der oben genannten Zweideutigkeit der Zuordnung beide Hälften gleichzeitig.

Wählt man eine feste Achse $\mathbf{r} \in \mathcal{K}^3$, so durchläuft ein Körper bei einer doppelten Volldrehung die Orientierungen $\{\mathbf{Q}_{\mathbf{r},\varphi} = \cos\frac{\varphi}{2} \cdot 1 + \sin\frac{\varphi}{2} \cdot \mathbf{r} \mid \varphi \in [0, 4\pi)\}$, welche einen Großkreis mit den Achsen $1, \mathbf{r} \in \mathcal{K}^4$ bilden. Um andere Achsen zu erhalten, sei eine Ausgangslage $\mathbf{P} \in \mathcal{K}^4$ vorgegeben, von der aus um $\mathbf{r}$ rotiert wird; wieder bildet $\{\mathbf{Q}_{\mathbf{r},\varphi} \cdot \mathbf{P} \mid \varphi \in [0, 4\pi)\}$ einen Großkreis, diesmal mit Achsen $\mathbf{P}$ und $\mathbf{r} \cdot \mathbf{P}$. Es gilt $\mathbf{P} \perp \mathbf{r} \cdot \mathbf{P}$ wegen

$$\mathbf{P}^{\mathrm{T}}(\mathbf{r} \cdot \mathbf{P}) = (p_0, \mathbf{p})^{\mathrm{T}}(-\mathbf{r}^{\mathrm{T}}\mathbf{p},\ p_0\mathbf{r} + \mathbf{r} \times \mathbf{p}) = -p_0\, \mathbf{r}^{\mathrm{T}}\mathbf{p} + p_0\, \mathbf{p}^{\mathrm{T}}\mathbf{r} + \underbrace{\mathbf{p}^{\mathrm{T}}(\mathbf{r} \times \mathbf{p})}_{\det[\mathbf{p},\mathbf{r},\mathbf{p}]=0} = 0$$

Umgekehrt entspricht einem Großkreis auf $\mathcal{K}^4$ eine Volldrehung eines Körpers, denn es läßt sich zu zwei vorgegebenen aufeinander senkrechten Großkreisachsen $\mathbf{P}, \mathbf{Q} \in \mathcal{K}^4$ eine Drehachse $\mathbf{r} \in \mathcal{K}^3$ finden, so daß $\mathbf{Q} = \mathbf{r} \cdot \mathbf{P}$; das ist deshalb so, weil

$$\mathbf{Q} \perp \mathbf{P} \iff q_0 p_0 + \mathbf{q}^{\mathrm{T}}\mathbf{p} = 0$$

und daher
$$\begin{aligned}\mathbf{Q} \cdot \mathbf{P}^* &= (q_0, \mathbf{q}) \cdot (p_0, -\mathbf{p}) \\ &= (q_0 p_0 + \mathbf{q}^{\mathrm{T}}\mathbf{p}, -q_0\mathbf{p} + p_0\mathbf{q} - \mathbf{q} \times \mathbf{p}) \\ &= (0, \mathbf{r}) \in \mathcal{K}^4\end{aligned}$$

Daraus folgt also

Satz 4.2:
Großkreise in $\mathcal{K}^4$ und die Lagen eines Körpers während Volldrehungen entsprechen einander.

Das bedeutet, daß die natürliche Bewegungsform der Drehung um eine feste Achse durch die Darstellung mittels Quaternionen auf eine Geodäte von $\mathcal{K}^4$ abgebildet wird, also auf die kürzeste Verbindung zweier Punkte auf dieser vierdimensionalen Kugel. Daher kann die heuristische Strategie für reine Rotationen anschaulich beschrieben werden als Übertragung des translatorischen Falles von der euklidischen in die sphärische Geometrie. Ein wichtiger Begriff ist dabei der des senkrechten Ausweichens, der noch näher untersucht werden muß.

Seien also ein Großkreis durch zwei Achsen $\mathbf{P}, \mathbf{Q} \in \mathcal{K}^4$, $\mathbf{P} \perp \mathbf{Q}$, und ein Punkt $\mathbf{K} \in \{\cos\varphi\mathbf{P} + \sin\varphi\mathbf{Q} \mid \varphi \in \mathbb{R}\}$ auf diesem gegeben. Jede weitere Achse $\mathbf{R} \in \mathcal{K}^4$, $\mathbf{R} \perp \mathbf{P}, \mathbf{Q}$, erzeugt nun zusammen mit $\mathbf{K}$ einen Großkreis senkrecht zum ursprünglichen; d.h. beide Großkreise schneiden sich in $\mathbf{K}$ und es gilt $\forall\, \psi \in \mathbb{R}:\ \mathbf{R} \perp \cos\psi\mathbf{P} + \sin\psi\mathbf{Q}$.

Umgekehrt seien nun zwei sich schneidende Großkreise gegeben durch den gemeinsamen Punkt $\mathbf{P}$ und je eine weitere Achse $\mathbf{Q} \perp \mathbf{P}$ und $\mathbf{R} \perp \mathbf{P}$. Diesen entsprechen Drehachsen $\mathbf{q}, \mathbf{r} \in \mathcal{K}^3$ mit $\mathbf{Q} = \mathbf{q} \cdot \mathbf{P}$, $\mathbf{R} = \mathbf{r} \cdot \mathbf{P}$. Was heißt es nun, daß zwei Kreise senkrecht aufeinander stehen? Bei glatten Kurven, welche sich in einem Punkt schneiden, wird man als Kriterium die Ableitungen im Schnittpunkt heranziehen. Betrachtet man die Großkreise als Kurven im $\mathbb{R}^4$, die gegeben sind durch die Funktionen

$\varphi \mapsto \cos\varphi\mathbf{P} + \sin\varphi\mathbf{Q}$ bzw. $\varphi \mapsto \cos\varphi\mathbf{P} + \sin\varphi\mathbf{R}$, so ist deren Ableitung (nach φ) im Schnittpunkt $\mathbf{P}$ gerade $\mathbf{Q}$ bzw. $\mathbf{R}$. Es gilt $\mathbf{Q} \perp \mathbf{R} \iff \mathbf{q}\cdot\mathbf{P} \perp \mathbf{r}\cdot\mathbf{P} \iff \mathbf{q} \perp \mathbf{r}$.

Satz 4.3:
Großkreise stehen senkrecht aufeinander genau dann, wenn die korrespondierenden Drehachsen senkrecht aufeinander stehen.

Beweis:
Mit obigen Bezeichnungen gilt

$$\begin{aligned}
\mathbf{Q}^{\mathrm{T}}\mathbf{R} &= (\mathbf{q}\cdot\mathbf{P})^{\mathrm{T}}(\mathbf{r}\cdot\mathbf{P}) \\
&= (-\mathbf{q}^{\mathrm{T}}\mathbf{p},\ p_0\mathbf{q}+\mathbf{q}\times\mathbf{p})^{\mathrm{T}}(-\mathbf{r}^{\mathrm{T}}\mathbf{p},\ p_0\mathbf{r}+\mathbf{r}\times\mathbf{p}) \\
&= \mathbf{q}^{\mathrm{T}}\mathbf{p}\ \mathbf{r}^{\mathrm{T}}\mathbf{p} + p_0^2\mathbf{q}^{\mathrm{T}}\mathbf{r} + p_0(\mathbf{q}^{\mathrm{T}}(\mathbf{r}\times\mathbf{p}) + (\mathbf{q}\times\mathbf{p})^{\mathrm{T}}\mathbf{r}) + \underbrace{(\mathbf{q}\times\mathbf{p})^{\mathrm{T}}(\mathbf{r}\times\mathbf{p})}_{\mathbf{q}^{\mathrm{T}}\mathbf{r}\ \mathbf{p}^{\mathrm{T}}\mathbf{p}-\mathbf{q}^{\mathrm{T}}\mathbf{p}\ \mathbf{r}^{\mathrm{T}}\mathbf{p}} \\
&= \underbrace{(p_0^2+\mathbf{p}^{\mathrm{T}}\mathbf{p})}_{|\mathbf{P}|^2=1}\mathbf{q}^{\mathrm{T}}\mathbf{r} + p_0\underbrace{(\det[\mathbf{q},\mathbf{r},\mathbf{p}] + \det[\mathbf{r},\mathbf{q},\mathbf{p}])}_{0} \\
&= \mathbf{q}^{\mathrm{T}}\mathbf{r}
\end{aligned}$$

■

Damit läßt sich die heuristische Bewegungsplanungsstrategie für den Fall eines im Raum frei rotierenden Körpers anschaulich beschreiben. Die vorgegebenen Anfangs- und Endorientierungen werden abgebildet auf unitäre Quaternionen. Als direkter Weg von Start zu Ziel wird jeweils der kürzere Großkreisbogen angesehen, der beide Punkte enthält; von diesem wird der Schnitt mit dem Konfigurationsraum ausgerechnet. Falls eine Kollision auftritt, so wird dieser senkrecht ausgewichen, d.h. auf einem Großkreis durch einen Kollisionspunkt und mit einer Achse senkrecht zu Start und Ziel. Ausweichpunkte auf jenem Großkreis werden dann in rekursiver Weise angesteuert, um so einen Pfad von der Anfangs- zur Endorientierung zu finden.

4.1.3 Warum eigentlich Quaternionen?

Wir möchten hier noch kurz erläutern, warum wir uns bei der Darstellung von Orientierungen für Quaternionen entschieden haben. Ziel war es, die heuristische Strategie ähnlich anschaulich wie bei der Bewegung eines Punktes im zwei- oder dreidimensionalen Raum zu beschreiben; d.h. durch Begriffe wie „direkter Weg“ und „senkrecht dazu ausweichen“.

Es mag naheliegend erscheinen, einfach die orthonormalen Matrizen mit Determinante +1 zu verwenden, die zuvor schon als Rotationsmatrizen beschrieben wurden. Diese bilden die spezielle orthogonale Gruppe $SO(3)$, ähnlich der Gruppe der unitären Quaternionen. Man kann $SO(3)$ als Teilmenge des $\mathbb{R}^9$ darstellen; aber da nur sechs der neun Elemente zur Festlegung einer orthonormalen Matrix wirklich nötig sind, läßt sich die Menge der Rotationsmatrizen auch in den $\mathbb{R}^6$ einbetten. Die erlaubten Rotationen eines frei beweglichen Körpers bilden auf diese Weise eine kompakte Menge im $\mathbb{R}^6$, die jedoch nicht einfach anschaulich zu beschreiben ist. Auch ist nicht ohne Weiteres erkennbar, wie die Geodäten in einer Geometrie auf $SO(3)$ aussehen; daher fällt es schwer, von der „direkten Verbindung“ zweier Rotationsmatrizen zu sprechen.

Man kann nun auch Rotationen mit nur drei reellen Parametern beschreiben, nämlich z.B. durch Angabe von Drehwinkeln bezüglich gewisser Achsen, um die der Körper nacheinander rotiert. Euler- oder Fixed-Winkel stellen solche bekannten Möglichkeiten dar. Bei dieser Abbildung von $SO(3)$ nach $\mathbb{R}^3$ geht jedoch die Struktur der orthogonalen Gruppe verloren. Zum einen ist Zuordnung von Winkeln zu Rotationsmatrizen nicht injektiv, zum anderen ist die zyklische Struktur der Rotationen in den drei getrennten Winkeln nur noch schwer zu erkennen. Als „direkten Weg“ kann man sicherlich eine Gerade im $\mathbb{R}^3$ verwenden, doch ergibt dies für die Kollisionsrechnung sehr schwer zu handhabende Wege in $SO(3)$. Vor allem aber bleibt der Begriff des „senkrechtes Ausweichens“ unbefriedigend, da der Bezug zu den Drehungen nicht so offensichtlich ist wie bei den Quaternionen (vgl. Satz 4.3).

4.2 Schnitt mit Konfigurationsraum

Wir haben gesehen, wie die Orientierung eines im Raum frei beweglichen Körpers repräsentiert werden kann mittels Quaternionen. Es wurde auch eine Vorgehensweise skizziert, Hindernissen bei rotatorischen Bewegungen auszuweichen. Ehe wir zu einer heuristischen Strategie für translatorische *und* rotatorische Freiheitsgrade gelangen, muß aber noch eine wichtige Grundlage geschaffen werden, nämlich die Erkennung der kollisionsfreien bzw. unfreien Abschnitte einer Rotation. Dies entspricht, wie im Abschnitt über sphärische Geometrie gezeigt wurde, dem Schnitt einer Geodäte mit den Konfigurationsraumhindernissen. Eine solche Drehung wird vom Planungsalgorithmus nachgefragt unter Angabe eine Ausgangslage, einer Drehachse und eines Winkels. Bekannt sind ferner ein bewegtes Objekt und eine Menge feststehender Hindernisse im Objektraum.

Bemerkenswert ist dabei, daß Rotationen für die Planung des Weges zwar als Quaternionen repräsentiert werden, weil die Wegesuche so leicht zu beschreiben ist, aber für die konkrete Kollisionsrechnung eine Darstellung mit Achse und Winkel benutzt wird. Diese ist den Quaternionen sehr verwandt, die Achse und Winkel lediglich ein wenig „kodiert“ darstellen, und führt auf eine wiederum sehr anschauliche Problemstellung beim Test auf unfreie Positionen, die recht einfach zu lösen ist. Der Vorteil der Quaternionen ist dagegen, daß man auf der Planungsebene leichter den Überblick behält.

Faßt man nun den Weg des Objekts während der Drehung als Abbildung $\rho : [0,1] \to \mathcal{K}^4$ auf, wobei $\rho(0)$ die Anfangs- und $\rho(1)$ die Endlage ist, so liefert die Kollisionsrechnung jene Zeitpunkte der Rotation, zu denen das bewegte Objekt mit einem Hindernis kollidiert. Dieses Ergebnis kann man durch eine Folge von Intervallen beschreiben.

Um eine effiziente Kollisionsrechnung zu erlauben, beschränken wir die erlaubten Körper wieder auf Polyeder. Dadurch werden die zu lösenden Ungleichungen später einen niedrigen Grad haben, denn die Oberfläche eines Polyeders läßt sich einfach beschreiben.

Die Frage nach Kollisionen des bewegten Polyeders mit den Hindernissen wird zurückgeführt auf das vom translatorischen Fall bekannte logische Prädikat für die Überlappung zweier Polyeder.

Dazu wird zunächst eine abkürzende Schreibweise eingeführt. Sei $\mathbf{r} \in \mathbb{R}^3$, $|\mathbf{r}| = 1$, wie zuvor auch eine Drehachse und $\varphi \in \mathbb{R}$ ein Winkel. Für ein $\mathbf{x} \in \mathbb{R}^3$ bezeichnet nun $\mathbf{x}_\varphi := R_{\mathbf{r},\varphi}(\mathbf{x})$ den bzgl. $\mathbf{r}$ um φ rotierten Punkt.

Es wird davon ausgegangen, daß die Objekte in ihrer Ausgangslage so beschrieben sind, daß die Drehachse durch den Ursprung verlaufen kann. Das bewegte Polyeder P drehe sich um $0 \leq \varphi_{Max} < 2\pi$ bzgl. $\mathbf{r}$, Q sei ein festes Hindernispolyeder. Eine Bewegung ist damit gegeben als stetige Zuordnung $\Phi : [0, 1] \to \mathbb{R}$, $t \mapsto \Phi(t) := t\varphi_{Max}$. Demnach entspricht der Zeitpunkt $t = 0$ wieder dem Anfang, $t = 1$ dem Ende der Bewegung; P befindet sich zur Zeit t in der Position $R_{\mathbf{r},\Phi(t)}(P)$. Analog zur Translation gilt auch hier, daß eine Rotation von Q bei festem P als Drehung von P in Gegenrichtung bei unbewegtem Q aufgefaßt werden kann. Aus diesem Grund wird von den Kollisionstests Punkt-Polyeder und Kante-Fläche nur eine Form beschrieben, bei der sich der Punkt bzw. die Kante bewegen.

Zuerst soll wieder die Überprüfung, ob ein Punkt in einem Polyeder liegt, an die Verhältnisse bei Drehungen angepaßt werden. Die Idee ist die gleiche, die schon bei Translationen verwendet wurde. Man wähle einen geeigneten Punkt $\mathbf{v}$ von P und betrachte den Kreisbogen, den er bei der Bewegung von P überstreicht; dessen Schnitte mit den Flächen von Q werden berechnet und nach ihrem Winkel auf dem Kreisbogen sortiert. Falls Flächen nicht orientiert sind, lassen sich freie und unfreie Positionen von $\mathbf{v}$ so noch nicht unterscheiden, da unklar ist, ob der Punkt das Polyeder Q betritt oder verläßt; zudem kann der Kreisbogen vollständig in Q liegen. Es muß aber lediglich von einem beliebigen Punkt auf dem Kreisbogen mittels eines Strahls in irgendeine Richtung entschieden werden, ob er innerhalb von Q liegt; die Information für den Rest des Kreisbogens ergibt sich dann durch den Wechsel von Innen und Außen bei jedem Schnittpunkt. Eine Entartung kann auch hier beim Schnitt mit Ecken und Kanten auftreten, dann wird ein anderer Punkt $\mathbf{v}$ gewählt. Geeignete Punkte lassen sich stets in einem Dreieck auf der Oberfläche von P finden, das nicht senkrecht zur Drehachse liegt; dann können wieder nicht alle inneren Punkte bei ihrer Kreisbahn Ecken oder Kanten von Q treffen.

Als nächstes wird nun die Durchdringung der Polyederoberflächen überprüft, wiederum durch Auswertung des Prädikats $S(l_{\mathbf{p},\mathbf{q}}, f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}})$ und darauf aufbauend durch $S(P, Q)$. Gesucht ist dabei für ein Kante-Fläche-Paar die Menge

$$\begin{aligned} \mathcal{P} &:= \{0 \leq \varphi < 2\pi \mid S(R_{\mathbf{r},\varphi}(l_{\mathbf{p},\mathbf{q}}), f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}})\} \\ &\subseteq \{0 \leq \varphi < 2\pi \mid R_{\mathbf{r},\varphi}(l_{\mathbf{p},\mathbf{q}}) \cap f_{\mathbf{v}_0,\ldots,\mathbf{v}_{k-1}} \neq \emptyset\} \end{aligned}$$

Diese Menge kann dann noch auf $[0, \varphi_{Max}]$ eingeschränkt werden, da nur diese Winkel bei der Drehung durchlaufen werden.

Mit obiger Notation gilt $R_{\mathbf{r},\varphi}(l_{\mathbf{p},\mathbf{q}}) = l_{\mathbf{p}_\varphi,\mathbf{q}_\varphi}$, was anschaulich bedeutet, daß das rotierte Liniensegment durch die rotierten Endpunkte festgelegt wird. Das Prädikat $S(l_{\mathbf{p},\mathbf{q}}, f)$ besteht aus zwei Arten von Ungleichungen, wie wir schon bei Translationen gesehen haben. Für diese beiden wird nun angegeben, wie man ihre Lösung bei Rotationen berechnet; dabei steht $\mathcal{R}$ wieder für ein Vergleichssymbol.

1. $\mathbf{n}^{\mathrm{T}}\mathbf{x}_\varphi - c \;\mathcal{R}\; 0$

   Von dieser Art sind die Abfragen, auf welcher Seite einer Ebene ein Punkt liegt ($\mathbf{n}^{\mathrm{T}}\mathbf{p} - n_0 \leq 0$), bzw. ob zwei Vektoren gleichgerichtet sind ($\mathbf{n}^{\mathrm{T}}(\mathbf{q} - \mathbf{p}) > 0$). Während der Drehung bleibt f und damit $\mathbf{n}$ fest, $\mathbf{p}$ und $\mathbf{q}$ ändern sich; dabei gilt $\mathbf{q}_\varphi - \mathbf{p}_\varphi = (\mathbf{q} - \mathbf{p})_\varphi$. Zur Anwendung einer Rotation auf einen Punkt sei nochmals auf Gleichung 4.1, Seite 53, verwiesen.

$$\begin{array}{lrcc} & \mathbf{n}^{\mathrm{T}}\mathbf{x}_\varphi - c & \mathcal{R} & 0 \\ \iff & \mathbf{n}^{\mathrm{T}}\Big((1-\cos\varphi)\mathbf{r}^{\mathrm{T}}\mathbf{x}\mathbf{r} + \cos\varphi\mathbf{x} + \sin\varphi\mathbf{r}\times\mathbf{x}\Big) - c & \mathcal{R} & 0 \\ \iff & \underbrace{(\mathbf{n}^{\mathrm{T}}\mathbf{x} - \mathbf{n}^{\mathrm{T}}\mathbf{r}\mathbf{r}^{\mathrm{T}}\mathbf{x})}_{\alpha}\cos\varphi + \underbrace{(\mathbf{n}^{\mathrm{T}}\mathbf{r}\times\mathbf{x})}_{\beta}\sin\varphi + \underbrace{(\mathbf{n}^{\mathrm{T}}\mathbf{r}\mathbf{r}^{\mathrm{T}}\mathbf{x} - c)}_{\gamma} & \mathcal{R} & 0 \end{array}$$

   Damit ist der ursprüngliche Test zurückgeführt auf eine Ungleichung der Form $\alpha\cos\varphi + \beta\sin\varphi + \gamma \;\mathcal{R}\; 0$. Die entsprechende Gleichung läßt sich geometrisch deuten als Schnitt einer Geraden $\alpha x + \beta y + \gamma = 0$ mit dem Einheitskreis; für einen Schnittpunkt gilt $x = \cos\varphi$ und $y = \sin\varphi$. Abhängig von der Relation $\mathcal{R}$ kann die allgemeine Form demnach z.B. auch den Schnitt einer Halbebene mit dem Einheitskreis beschreiben.

   Der Winkel $\varphi = \pi$ ist Nullstelle der Gleichung genau dann, wenn $-\alpha + \gamma = 0$, d.h. daß die Gerade durch den Punkt $(-1, 0)$ verläuft. Ein zweiter Schnittpunkt existiert genau dann, wenn $\beta \neq 0$; sonst ist die Gerade eine Senkrechte und tangiert den Kreis nur, dabei sei $\alpha \neq 0$, damit die Gerade nicht zur Ebene entartet. Berechnen läßt sich der Schnittpunkt mit nachfolgend beschriebener Vorgehensweise, die in diesem Sonderfall zu der linearen Gleichung $2\beta t + 2\alpha = 0$ führt.

   Für Winkel $-\pi < \varphi < \pi$ löst man die Ungleichung nun elegant über die entsprechende Gleichung und diese durch die Substitution $t := \tan\frac{\varphi}{2}$. Es ergibt sich $\cos\varphi = \frac{1-t^2}{1+t^2}$, $\sin\varphi = \frac{2t}{1+t^2}$ und nach Multiplikation mit $1 + t^2$

$$\alpha(1-t^2) + \beta 2t + \gamma(1+t^2) = 0 \iff (\gamma - \alpha)t^2 + 2\beta t + (\alpha + \gamma) = 0$$

   Mit $\alpha \neq \gamma$ führt die bekannte Lösungsformel für quadratische Gleichungen jetzt zur Bedingung $\alpha^2 + \beta^2 \geq \gamma^2$ für die Lösbarkeit und dann zu den Nullstellen

$$t_{1,2} = \frac{\beta \mp \sqrt{\alpha^2 + \beta^2 - \gamma^2}}{\alpha - \gamma}$$

Mittels $\varphi = 2 \arctan t$ überführt man diese in Winkel.

Hat man unter sorgfältiger Beachtung aller Sonderfälle die Lösungen der Gleichung als Winkel bestimmt, so kann man daraus durch Berücksichtigung von $\mathcal{R}$ auf die Lösungsmenge der allgemeinen Form der Ungleichung schließen. Die Nullstellen der Gleichung teilen die Kreislinie in Intervalle, aus denen $\mathcal{R}$ die Lösungsmenge auswählt. Diese kann durch maximal ein Intervall auf der Kreislinie beschrieben werden, da man $\neq$ als Relation $\mathcal{R}$ ausschließt. Mittels der Transformation der Winkelwerte ins Intervall $[0, 2\pi)$ erhält man letztendlich die gesuchte Darstellung der Lösungsmenge.

2. $\det[\mathbf{x}_\varphi, \mathbf{w}, \mathbf{p}_\varphi - \mathbf{v}] \;\mathcal{R}\; 0$

   Dies rührt von der im Prädikat verwendeten Abfrage $\det[\mathbf{q}-\mathbf{p}, \mathbf{v}_{i+1}-\mathbf{v}_i, \mathbf{p}-\mathbf{v}_i] \geq 0$ her, wobei $\mathbf{p}$ und $\mathbf{q}$ sich durch die Rotation bewegen, während die Fläche und damit ihre Eckpunkte $\mathbf{v}_i$ und $\mathbf{v}_{i+1}$ fest bleiben. Man kann hoffen, diesen Fall analog zur Vorgehensweise bei Translationen auf dieselbe Ungleichung zu reduzieren wie den ersten Fall. Die Determinante zerlege man jetzt mittels

   $$\det[\mathbf{x}_\varphi, \mathbf{w}, \mathbf{p}_\varphi - \mathbf{v}] = \det[\mathbf{x}_\varphi, \mathbf{w}, \mathbf{p}_\varphi] - \det[\mathbf{x}_\varphi, \mathbf{w}, \mathbf{v}]$$

   in zwei Teile. Letztere stellen Spatprodukte dar, d.h. geben das Volumen eines von den drei Vektoren aufgespannten Spats an; dieses ist natürlich invariant unter Rotationen. Durch Anwendung der Umkehrrotation $R_{\mathbf{r},-\varphi}$ auf dieses Spat erhält man daher

   $$\det[\mathbf{x}_\varphi, \mathbf{w}, \mathbf{p}_\varphi] = \det[\mathbf{x}, \mathbf{w}_{-\varphi}, \mathbf{p}] = \det[\mathbf{w}_{-\varphi}, \mathbf{p}, \mathbf{x}]$$

   Man hat also nur Determinanten der Form $\det[\mathbf{x}_\varphi, \mathbf{v}, \mathbf{w}] = (\mathbf{v} \times \mathbf{w})^{\mathrm{T}} \mathbf{x}_\varphi = \mathbf{s}^{\mathrm{T}} \mathbf{x}_\varphi$ zu betrachten. Dabei stehen $\mathbf{v}$ und $\mathbf{w}$ und damit $\mathbf{s}$ für feste Punkte, $\mathbf{x}$ dagegen rotiert um die Achse durch den Nullpunkt mit Richtungsvektor $\mathbf{r}$ um den Winkel φ. Durch Anwendung von Gleichung 4.1 ergibt sich unmittelbar

   $$\begin{aligned} \mathbf{s}^{\mathrm{T}} \mathbf{x}_\varphi &= \mathbf{s}^{\mathrm{T}} \left((1 - \cos\varphi) \mathbf{r}^{\mathrm{T}} \mathbf{x} \mathbf{r} + \cos\varphi \mathbf{x} + \sin\varphi \mathbf{r} \times \mathbf{x} \right) \\ &= (\mathbf{s}^{\mathrm{T}} \mathbf{x} - \mathbf{s}^{\mathrm{T}} \mathbf{r} \mathbf{r}^{\mathrm{T}} \mathbf{x}) \cos\varphi + (\mathbf{s}^{\mathrm{T}} (\mathbf{r} \times \mathbf{x})) \sin\varphi + \mathbf{s}^{\mathrm{T}} \mathbf{r} \mathbf{r}^{\mathrm{T}} \mathbf{x} \\ &= \alpha \cos\varphi + \beta \sin\varphi + \gamma \end{aligned}$$

   Eine einzelne Determinante $\det[\mathbf{x}_\varphi, \mathbf{v}, \mathbf{w}]$ läßt sich also als Cosinus-Sinus-Term schreiben, damit ergibt sich für die Differenz solcher Spatprodukte die gleiche Form und für die ursprüngliche Abfrage eine Ungleichung der weiter oben schon behandelten Art

   $$\alpha \cos\varphi + \beta \sin\varphi + \gamma \;\mathcal{R}\; 0$$

Das Prädikat $S(l_{\mathbf{p},\mathbf{q}}, f)$ und damit auch $S(P, Q)$ kann aus den oben genannten Gründen auf die gleiche Art ausgewertet werden wie bei Translationen, d.h. mit den atomaren Ungleichungen verbindet man die Lösungsintervalle der oben beschriebenen

Cosinus-Sinus-Ungleichung und verwaltet als Zwischenergebnisse Intervallisten. Man erhält damit die Menge aller Winkel während der Rotation, bei denen die Oberflächen der betrachteten Polyeder sich schneiden. Zudem wurde erläutert, wie man feststellt, wann ein Polyeder bei der Drehung ganz in einem anderen verschwindet. Damit lassen sich also insgesamt für eine gegebene Rotation alle unfreien Abschnitte bestimmen und als Folge von Intervallen repräsentieren. Die Laufzeit für eine Kollisionsrechnung ergibt sich analog zum translatorischen Fall zu $O(n \log n)$ mit $n = |P||Q|$.

Genau wie bei Verschiebungen kann auch bei Drehungen durch die Verwendung von Hüllkörpern eine deutliche Beschleunigung dieser Berechnungen erzielt werden. Es werden sowohl Kugeln benutzt als auch Isoboxen in einem Zylinderkoordinatensystem. Damit die Invarianz bezüglich der Rotation gewahrt ist, werden nicht die kartesischen Raumachsen sondern die Parameter Winkel, Höhe und Abstand zur Rotationsachse zur Beschreibung der Hüllquader eingesetzt. Dabei entspricht der Drehwinkel in seiner Bedeutung der Koordinate in Verschiebungsrichtung, und die bei Translationen erläuterten Techniken lassen sich auch hier anwenden.

4.3 Frei beweglicher Körper im Raum

Nachdem nun alle wichtigen Grundlagen gelegt sind, soll eine heuristische Strategie zur Wegesuche für einen im dreidimensionalen Raum frei beweglichen starren Körper zwischen Hindernissen beschrieben werden. Alle Objekte seien dabei als Polyeder repräsentiert, ferner sei die Plazierung aller Hindernisse sowie eine Start- und Zielposition des beweglichen Körpers bekannt. Gesucht ist dann ein kollisionsfreier Weg vom Start zum Ziel für das bewegte Polyeder, der stückweise aus Verschiebungen und Änderungen der Orientierung besteht. Wie schon zuvor darf die Suchstrategie nur Anfragen nach dem Schnitt einer Geodäte mit dem Konfigurationsraum stellen; translatorische und rotatorische Freiheitsgrade werden strikt getrennt behandelt. Auf diese Weise wird keine explizite Darstellung des Konfigurationsraums benötigt, vielmehr reicht die implizite Definition mittels des Objektraums aus, um die gewünschten lokalen Informationen effizient zu liefern.

Der Planungsalgorithmus entspricht in seiner Struktur dem bereits in Abschnitt 2.2, Seite 31, vorgestellten für den Fall mehrerer getrennter Freiheitsgrade. Ähnlich wie dort wird Buch darüber geführt, mit welcher Bewegungsart ein bestimmter Abschnitt des Weges beginnen soll, damit auf dem Weg zu einem Ausweichpunkt und von dort zum Ziel nicht noch einmal die bereits bekannte, unfreie Position anvisiert wird, welche das Ausweichen verursachte. Man kann sich das anschaulich so vorstellen, daß die Position des Körpers durch je einen Punkt im $\mathbb{R}^3$ und $\mathcal{K}^4$ repräsentiert wird. Die Wegesuchstrategie bewegt nun abwechselnd den Punkt im euklidischen Raum und den auf der Kugeloberfläche, wobei Geraden bzw. Großkreise benutzt werden. Von diesen Wegstücken werden die Abschnitte ermittelt, die innerhalb von Konfigurationsraumhindernissen liegen, also verbotenen Positionen entsprechen. Wenn der betrachtete Weg frei ist, zieht der Punkt bis zur Zielposition, sonst wird dem ersten unfreien Abschnitt ausgewichen, genauer gesagt dessen Mittelpunkt. Auf diese Weise ergibt sich ein Umwegpunkt, der vom Start aus angesteuert wird und von dem aus der Weg zum Ziel gesucht wird; diese beiden Probleme werden rekursiv gelöst. Abgesehen von der Kugeloberfläche, mit der die Orientierungen dargestellt werden, ist das analog zum rein translatorischen Fall.

Sehr viel interessanter als die Planungsebene ist jedoch der Teil der Wegesuche, der die Umwegpunkte auswählt, über die der Weg verlaufen soll. Die Idee dabei ist, daß z.B. der Punkt auf seinem Weg im $\mathbb{R}^3$ von der Start- zur Zielposition jede Ebene durchdringen muß, die Start und Ziel trennt. Wenn nun der direkte Weg nicht kollisionsfrei ist, so erfüllt sicherlich eine Ebene durch irgendeinen unfreien Punkt des Weges und senkrecht zur Richtung desselben obige Bedingungen. Demnach muß „nur noch" ein Punkt in jener Ebene gefunden werden, durch den es einen Weg vom Start zum Ziel gibt. Für den Punkt auf der Kugeloberfläche stellt sich die Situation ähnlich dar, nur läßt sie sich schwer anschaulich beschreiben, da die Fläche selbst dreidimensional ist.

Solche Ausweichpositionen werden ermittelt, indem verschiedene Ausweichbewegungen bei der Kollisionsrechnung nachgefragt und freie Intervalle auf diesen Wegen gesammelt werden. Die Auswahl der Umwegpunkte beschränkt sich dann auf die Mittelpunkte dieser Freiintervalle, diese werden noch gemäß einer Bewertung angeordnet und eventuell auf eine feste Anzahl reduziert, z.B. auf die beiden besten. Wenn die Bewertungsfunktion gut ist, dann werden vielversprechende Wege zuerst ausprobiert, ungünstiger erscheinende dagegen nur, wenn es sich nicht vermeiden ließ. Die Beschränkung der Zahl der Ausweichpunkte sorgt für einen kleinen Knotengrad im Rekursionsbaum und hilft damit dem Backtracking.

Welche Ausweichbewegungen kommen nun in Frage? Prinzipiell sind dies beliebige Rotationen und Translationen, die lediglich „orthogonal" zu der Bewegung sein müssen, welche die Kollision verursachte. Falls letzteres eine Translation in Richtung $\mathbf{v}$ war, so sind Verschiebungen nur noch senkrecht zu $\mathbf{v}$ erlaubt, Drehungen dagegen uneingeschränkt. Falls eine Rotation um die Achse $\mathbf{v}$ auf ein Hindernis stieß, so werden die erlaubten Drehachsen auf solche senkrecht zu $\mathbf{v}$ beschränkt, wogegen alle Translationsrichtungen zugelassen bleiben. Es wurde ja gezeigt, daß Orthogonalität von Großkreisen und damit den Wegen bei Rotation und Orthogonalität von Drehachsen gleichbedeutend sind.

Unter den erlaubten Ausweichbewegungen werden nun entweder einige erwürfelt oder besonders interessant erscheinende ausgezeichnet, z.B. durch ein Orthogonalsystem von Richtungen. Als Startpunkt kommt dabei nicht nur eine der ermittelten unfreien Plazierungen in Betracht, sondern auch solche Positionen, die daraus durch eine erlaubte Bewegung hervorgehen. Konkret kann das Objekt also erst ein Stück verschoben werden, ehe eine Volldrehung als Ausweichbewegung nachgerechnet wird.

All diese Möglichkeiten dürfen aber nicht darüber hinweg täuschen, daß die Wahl des richtigen Umwegpunktes in schwierigen Szenen für die Heuristik mehr eine Glückssache ist; in leichten Szenen dagegen führt fast jeder Umwegpunkt zum Ziel. Daher sollte eine Arbeit zur weiteren Verbesserung dieser heuristischen Wegesuche bei der

Wahl der Umwegpunkte ansetzen, sie bietet einiges Potential und ist entscheidend für die Güte der gefundenen Wege und die dabei verbrauchte Zeit.

Anhang A

Die zweidimensionale Implementierung

A.1 Einige ausgewählte Beispiele

Die heuristische Strategie zur Wegesuche in der Ebene wurde im Rahmen eines Fortgeschrittenenpraktikums vom Autor implementiert. Es handelt sich um ein Programm mit graphischer Oberfläche, das Bewegungen für einen Punkt zwischen Kreisscheiben planen kann. Eine Aufgabenstellung kann sowohl interaktiv als auch automatisch durch Pseudozufallszahlen erzeugt werden, zur Lösung stehen die drei beschriebenen Rekursionsstrategien zur Verfügung. Als besondere Option läßt sich die Zahl der jeweils betrachteten Umwegpunkte auf maximal Zwei beschränken; ferner können Wege erzeugt werden, die stückweise parallel zu den Koordinatenachsen verlaufen.

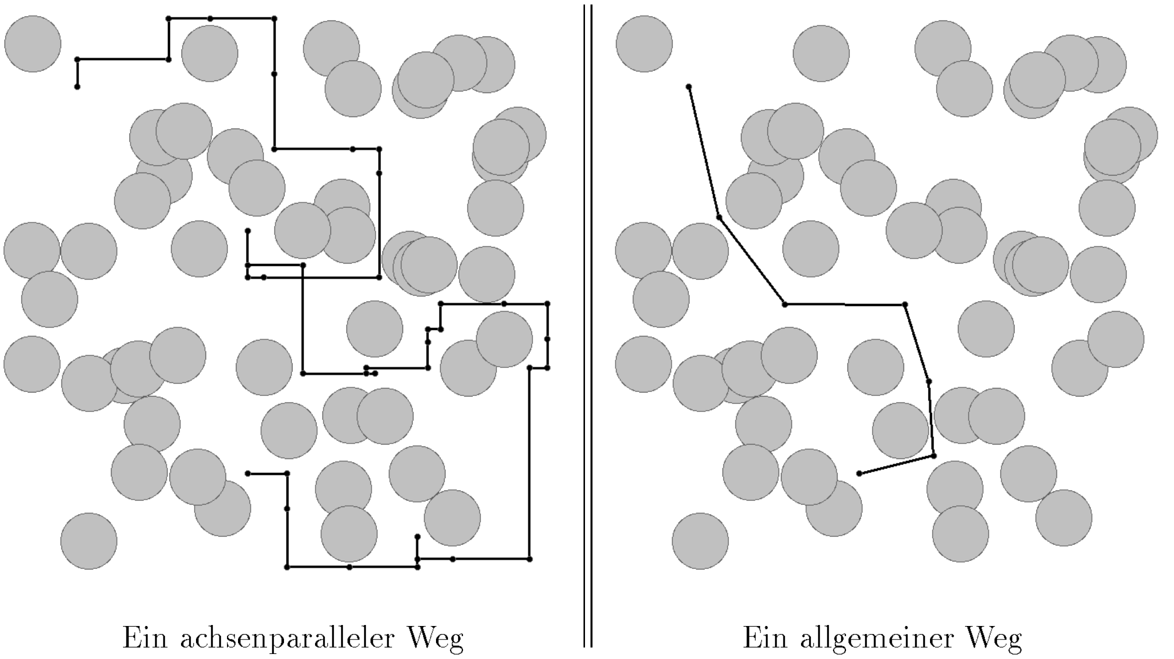
Da die von unserem Verfahren erzeugten Wege, insbesondere bei deutlich zu hoch gewählter Rekursionstiefe, oft nicht elegant genug verlaufen, steht noch eine Routine zur Verkürzung errechneter Wege zur Verfügung. Dabei wird ein Graph gebildet, dessen Knoten die Ecken des momentanen Pfades sind; eine Kante verbindet Ecken, deren Direktverbindung das Hindernis nicht schneidet. In diesem Graphen wird der kürzeste Weg zwischen Start und Ziel bestimmt und ergibt so einen neuen, im allgemeinen sehr eleganten Pfad. Diese Nachbesserung geht sehr schnell und läßt sich auf höhere Dimensionen übertragen; dort nimmt dann auch das Planen eines Weges mehr Zeit in Anspruch als das anschließende Verkürzen.

Achsenparallele Wege

Im Abschnitt 2.2, Seite 31, wurde ein Algorithmus vorgestellt, der die erlaubten Richtungen eines Wegstücks einschränkt. Ein gutes Anwendungsbeispiel dafür findet sich in der Ebene, wenn man diese nicht mehr einheitlich betrachtet, sondern nach x- und y-Achse unterscheidet. Zwischen zwei gegebenen Punkten darf ein Weg nun nicht mehr direkt verlaufen, sondern nur stückweise parallel zu den Koordinatenachsen; die „Direktverbindung“ bildet dann einen Knick und repräsentiert die sogenannte Manhattan-

Distanz der Punkte. Man könnte sich dies z.B. auch als einen Plotter vorstellen, der seinen Stift zwar in jeder Achsrichtung bewegen kann, diese Bewegungen aber nicht überlagern darf.

Abbildung A.1 zeigt links, wie ein auf diese Weise geplanter Weg aussehen kann. Die Szene entstand durch automatische Verteilung gleichgroßer Kreischeiben im Einheitsquadrat, Ausgangs- und Zielpunkt wurden von Hand festgelegt. Der Pfad wurde mit maximal zwei Umwegpunkten und minimaler Rekursionstiefe geplant, letztere war sechs. Dabei wurden innerhalb von drei Sekunden 625 „intersect"-Aufrufe zur Planung benötigt.



Ein achsenparalleler Weg | Ein allgemeiner Weg

Abbildung A.1: Vergleich zweier Wege

Im Vergleich dazu zeigt Bild A.1 rechts einen Weg, der ohne diese Beschränkung der Richtung und mit Hilfe der Baumstruktur zum Auffinden eines rekursionsminimalen Pfades geplant wurde. Dabei genügt Rekursionstiefe drei, und nur 26 Schnittlinien wurden ausgerechnet; das Ergebnis erscheint ohne Verzögerung auf dem Bildschirm. Dies sollte die Auswirkungen derart starker Einschränkungen an die Bewegungsfreiheit eines Objekts verdeutlichen, zumindest im Hinblick auf dieses einfache heuristische Planungsverfahren.

Man beachte jedoch, daß eine Annäherung des allgemeinen Wegs durch Treppenstufen in x- und y-Richtung ebenfalls eine starke Erhöhung der erforderlichen Schnittlinien bewirkt; insofern ist der Nachteil achsenparalleler Planung nur natürlich. Die wahren Vorteile der Vorgehensweise werden dann deutlich, wenn überlagerte Bewegungen von der Kollisionsrechnung nur schwer verarbeitet werden können, die einzelnen, reinen Bewegungsarten dagegen sehr viel leichter.

Einfluß der Rekursionstiefe auf den Weg

Unser nächstes Beispiel zeigt eine Szene aus verschieden großen Kreisscheiben, die nur wenige enge Durchgänge lassen; dabei begrenzt ein äußerer Rand den Bewegungsspielraum für den Punkt zusätzlich. In dieser Situation ist nun zuerst ein Weg mit der vorgegebenen Rekursionstiefe 15 geplant worden; er ist in Abbildung A.2 links oben zu sehen und sehr verwickelt. Man beachte insbesondere, daß der Weg nicht nur sich selbst mehrfach überkreuzt, sondern in diesen Schleifen sogar Hindernisteile umrundet. Bei seiner Berechnung waren 13414 „intersect"-Aufrufe erforderlich, was geraume Zeit dauerte; ohne die Beschränkung auf zwei Umwegpunkte werden sogar 26985 Schnittlinien errechnet.

Im Gegensatz dazu ist ein Weg mit optimaler Rekursionstiefe von acht deutlich eleganter und auch schneller geplant, namentlich mit nur 684 Schnitten. Läßt man der Backtracking-Strategie wenig Spielraum, so müssen schlechte Entscheidungen zurückgenommen und geschickt gewählte Umwegpunkte bevorzugt werden. Andernfalls kann auch ein unglücklicher Anfang oft noch zu einer erfolgreichen Berechnung fortgesetzt werden, die überflüssigen Schleifen und Umwege bleiben aber in der Lösung sichtbar. Zudem wird, wie bereits erwähnt, in diesem Fall eine ungünstige Wahl einer Ausweichposition rasch bemerkt, denn es muß kein sehr tiefer Baum mehr vollständig durchsucht werden.

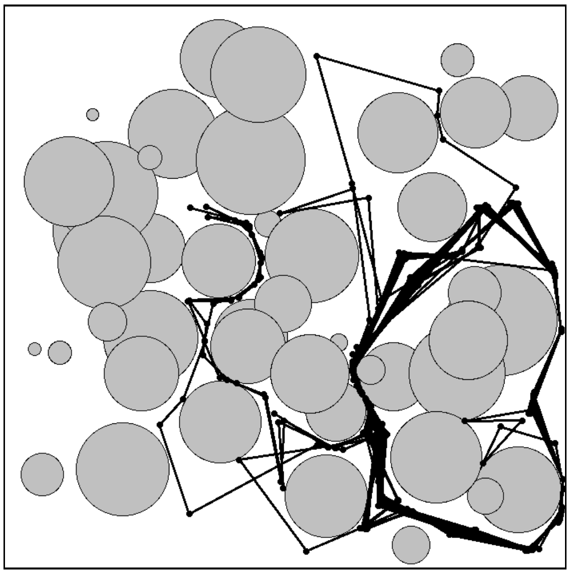
Selbst dieser schon recht gute Weg enthält aber noch einige unnötig scheinende Schleifen; diese werden dadurch verursacht, daß die Wahl der Grenzgeraden und der jeweiligen Umwegpunkte eben doch eine gewisse Einschränkung an die konstruierbaren Wege darstellt. Hier hilft eine Nachbearbeitung des Pfades, das Ergebnis der Verkürzung ist ebenfalls abgebildet.

A.2 Statistische Auswertung einiger Experimente

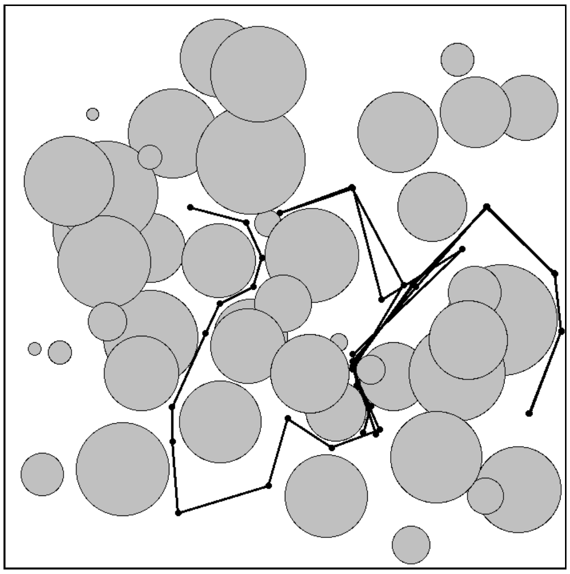
Die hohe Geschwindigkeit des Planungsalgorithmus für einfache Problemstellungen in der Ebene erlaubt es, eine große Zahl pseudozufälliger Bewegungsplanungsprobleme zu erzeugen und lösen zu lassen. Dabei können dann gewisse Details der Strategie verändert und ihr Einfluß auf die Effizienz untersucht werden, mit der eine solche Klasse von Szenen behandelt wird. Für die folgenden Experimente wurde jeweils die gleiche Folge von 10000 Aufgabenstellungen aus fünfzig gleich großen Kreisen im Einheitsquadrat produziert, welches diagonal von links oben nach rechts unten zu durchqueren war. Da alle Kreise innerhalb der Begrenzung lagen, waren alle Probleme lösbar; die Planungsstrategie aber wählt selbstverständlich nicht einfach den Weg auf dem Rand.

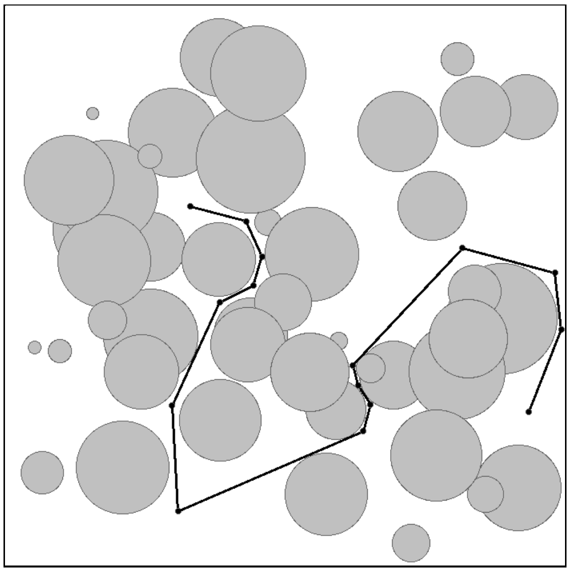
Bewertung der Umwegpunkte

Die folgende Tabelle A.1 zeigt für fünf verschiedene Werte von β jeweils die durchschnittlich und maximal benötigte Zahl von „intersect"-Aufrufen, mit denen das heuristische Verfahren die einzelnen Probleme löste. Ferner wird die ermittelte minimal



Links dargestellt ist der mit Rekursionstiefe 15 geplante Weg.





Oben sieht man den mit optimaler Tiefe erzeugten Weg, links das Ergebnis des Verkürzens.

Abbildung A.2: Einfluss der Rekursionstiefe

notwendige Rekursionstiefe vermerkt, wiederum als arithmetisches Mittel und Maximum; dabei wurde die schlichte Strategie verwendet, Rekursionstiefen von Null an auszuprobieren. Zusätzlich wurde jedes Experiment sowohl mit unbeschränkter Zahl von Umwegpunkten als auch mit maximal zwei je Grenzgerade ausgeführt.

Man bemerkt, daß die kleinstmögliche Rekursionstiefe in einer Szene nicht von β abhängt, sofern man alle Umwegpunkte betrachtet; der Parameter wirkt sich lediglich auf die Reihenfolge aus, in der diese Punkte betrachtet werden. Ein gute Bewertungsfunktion hilft hier Laufzeit zu sparen, indem die Strategie weniger oft in Sackgassen gerät.

Für den Fall $\beta = 2$ hatte das Verfahren bei Beschränkung auf maximal zwei Um-

β	maximal 2 Umwegpunkte				sämtliche Umwegpunkte			
	Rekursionstiefe		Schnittlinien		Rekursionstiefe		Schnittlinien	
	Max.	Mittel	Max.	Mittel	Max.	Mittel	Max.	Mittel
0	9	4.606	3362	194.4	6	3.812	3688	321.2
1/2	7	4.471	1039	167.0	6	3.812	4246	304.5
2/3	8	4.463	1937	167.6	6	3.812	4367	302.6
1	9	4.573	7655	194.8	6	3.812	3808	308.4
2	20	5.528	$2 \cdot 10^7$	$2 \cdot 10^4$	6	3.812	4545	339.8

Tabelle A.1: VARIATION VON β BEI MINIMALER REKURSIONSTIEFE

β	maximal 2 Umwegpunkte				sämtliche Umwegpunkte			
	Rekursionstiefe		Schnittlinien		Rekursionstiefe		Schnittlinien	
	Max.	Mittel	Max.	Mittel	Max.	Mittel	Max.	Mittel
1/2	7	4.471	557	94.0	6	3.812	3609	236.8
2/3	8	4.463	1084	93.9	6	3.812	3649	234.5
1	9	4.573	2939	106.2	6	3.812	3033	234.6

Tabelle A.2: VARIATION VON β BEI MINIMALER REKURSIONSTIEFE UND GESCHICKTER REKURSIONSSTRATEGIE

wegpunkte mit einigen Szenen große Schwierigkeiten, im Extremfall wurden bei Rekursionstiefe 20 über 21 Millionen Schnittlinien berechnet. Insgesamt lief dieses Experiment dann auch vier Tage und 11,5 Stunden, während bei den anderen lediglich Ausführungszeiten von wenigen Stunden zu beobachten waren.

Als Resumé ergibt sich, daß die Balance zwischen Abstand und Breite für diese Klasse von Bewertungsfunktionen bei $\beta \approx 2/3$ optimal erscheint. Ferner gibt es ein Trade-off zwischen Rekursionstiefe und Anzahl der benötigten Schnittlinien, wenn man die Zahl der betrachteten Umwegpunkte begrenzt.

Eine ganz ähnliche Versuchsreihe zeigt in Tabelle A.2 die Wirksamkeit der Baumstruktur, mit deren Hilfe ein rekursionsminimaler Pfad rascher bestimmt werden kann.

β	maximal 2 Umwegpunkte		sämtliche Umwegpunkte	
	Schnittlinien		Schnittlinien	
	Maximum	Mittel	Maximum	Mittel
0	3069	94.74	7306	149.2
2/5	1933	51.04	?	?
1/2	1793	47.26	4178	90.33
4/7	1802	46.41	3961	86.49
2/3	1837	45.97	3633	83.38
1	3159	72.19	2056	90.52

Tabelle A.3: VARIATION VON β BEI FESTER REKURSIONSTIEFE

Dabei werden Informationen über das Bewegungsplanungsproblem und seine Teile abgespeichert, z.B. die Kollisionsintervalle auf einer Direktverbindung oder die gewählten Umwegpunkte auf einer Grenzgeraden. Statt nun die Rekursionstiefe von Null an zu erhöhen und jedesmal diese Rechnungen erneut durchzuführen, wird auf Informationen der vorhergehenden Tiefe zurückgegriffen. Dabei werden hauptsächlich die Blätter der Baumstruktur durchforstet, um Wege mit einer um Eins erhöhten Tiefe zu suchen und den Baum zu vertiefen. Auf diese Weise lassen sich viele Aufrufe von „intersect“ einsparen; im Gegenzug steigt der Speicherbedarf an.

Eine letzte Versuchsreihe dieser Art zeigt Tabelle A.3, allerdings für den Fall einer fest gewählten Rekursionstiefe. Diese betrug entsprechend den Erfahrung aus obigen Experimenten neun bei Beschränkung auf zwei Umwegpunkten und sechs sonst. Der kritische Wert $\beta = 2$ wurde vorsichtshalber vermieden, die zugehörige Bewertungsfunktion ist schlichtweg unbrauchbar.

Wahl der Grenzgeraden

Eine interessante Frage bei der detaillierten Festlegung des Wegesuchverfahrens war, wo die Grenzgerade zwecks Suche nach Umwegpunkten errichtet werden soll. Zur Untersuchung dieser Frage diente eine Reihe von Experimenten, wie sie zuvor schon geschildert wurden; angesichts der bereits genannten Ergebnisse mit einem Parameter von $\beta = 2/3$ und der besten Rekursionsstrategie sowie einer Beschränkung auf maximal zwei Umwegpunkte. Betrachtet wurden jeweils Senkrechten, deren Fußpunkt auf der Direktverbindung $(\mathbf{a}, \mathbf{z})$ unterschiedlich gewählt wurde. Dabei wurden sowohl Mittelpunkte von Kollisionsintervallen als auch zufällige Punkte verwendet, außerdem der Anfang des ersten Kollisionsintervalls. Als Intervalle kamen das erste, mittlere und ein zufälliges in Frage; ferner wurde der am nächsten bei der Mitte von $(\mathbf{a}, \mathbf{z})$ gelegene Intervallmittelpunkt betrachtet.

	Rekursionstiefe		Schnittlinien	
Ort des Fußpunkts	Max.	Mittel	Max.	Mittel
Anfang des 1. Intervalls [a]	10	5.972	9086	547.6
zufällig auf dem 1. Intervall	8	4.612	806	106.8
Mitte des 1. Intervalls	8	4.463	1084	93.9
Mitte eines zufälligen Intervalls	7	4.026	440	64.9
Mitte des mittleren Intervalls	7	3.649	591	53.2
Intervallmitte nahe Mitte von $(\mathbf{a}, \mathbf{z})$	7	3.616	581	51.3
Mitte von $(\mathbf{a}, \mathbf{z})$	9	4.380	1359	89.6
zufällig auf $(\mathbf{a}, \mathbf{z})$	12	5.972	5375	211.0

[a] Wegen des unglücklichen Verhaltens dieser Strategie bei etlichen Beispielszenen mußte die Rekursionstiefe auf 10 beschränkt werden; an dieser Grenze scheiterte die Lösung von 173 der Szenen. Diese tragen zur Statistik der benötigten Schnittlinien bei, nicht aber zur Rekursionstiefe.

Tabelle A.4: Verschiedene Fusspunkte von Grenzgeraden

Die Ergebnisse sind in der Tabelle A.4 zu finden. Man erkennt, daß sich ein zufällig verteilter Punkt sehr ähnlich verhält wie der Erwartungswert der entsprechenden Verteilung; daher kann man die Strategie durchaus deterministisch implementieren. Wie bereits zuvor gesagt, zeigen sich darüber hinaus zwei Tendenzen: Intervallmittelpunkte und die Mitte von $(\mathbf{a}, \mathbf{z})$ sind zu bevorzugen, optimal ist eine Kombination aus beidem, nämlich der am nächsten zur Mitte gelegene Mittelpunkt.

Anhang B

Bewegung von Polyedern im Raum

Am Lehrstuhl von Prof. Hotz an der Universität des Saarlandes wurde von Elmar Schömer im Rahmen seiner Dissertation [Sch94] ein System zur **i**nteraktiven **g**raphischen **O**bjektmontage und **R**obotersimulation — kurz $\mathbb{ICOR}$ — entwickelt. Ziel davon ist es, Montagevorgänge mitsamt der beteiligten Teile und Werkzeuge im Computer zu simulieren, wobei ein Benutzer mit dem System interagiert, um die Bewegungen zu spezifizieren, zu analysieren und zu korrigieren. Dabei werden die einzelnen Gegenstände durch Polyeder modelliert, welche im Raum beliebig gedreht und verschoben werden können; einen Schwerpunkt bildet in diesem Zusammenhang die effiziente on-line-Kollisionserkennung.

Um die Arbeit mit einem solchen System zu erleichtern, kann man es durch ein Modul zur automatischen Bewegungsplanung erweitern. Dadurch muß ein Benutzer nur noch die gewünschte Endlage eines Objekts angeben, ein kollisionsfreier Weg dorthin wird vom Programm errechnet. Die in dieser Arbeit beschriebene heuristische Strategie zur Wegesuche wurde daher vom Autor in das $\mathbb{ICOR}$-System integriert; dabei können Bewegungen für einzelne Polyeder unter Ausnutzung von Translation und Rotation erzeugt werden. Ein so entstandener Ablauf läßt sich dann weiterverwenden und in einen Montageplan eingliedern. Wie auch schon im zweidimensionalen Fall sollen nun einige Beispiele von Bewegungsplanungsproblemen und ihren Lösungen gezeigt werden.

B.1 Einfügen eines Würfels in eine Schachtel

Die Aufgabenstellung besteht aus einem Würfel, welcher mit nur wenig Spielraum in eine quaderartige Schachtel paßt. Der zu planende Weg soll den Würfel nur durch Translationen aus einer Ausgangslage außerhalb der Schachtel an seine Endposition in deren Innerem bringen. In Abbildung B.1 sieht man den Würfel rechts hinter der Schachtel, etwas nach oben versetzt; das Wegesuchproblem wurde mit Rekursionstiefe fünf in Sekundenschnelle gelöst und ergab einen Pfad, der sich noch verkürzen ließ. Dieser Pfad wird zur Veranschaulichung durch einen Polygonzug im Raum repräsentiert. Weiterhin erkennt man den Bewegungsablauf anhand der Position des Würfels jeweils bei Richtungsänderungen, in vier Schritten erreicht er sein Ziel.

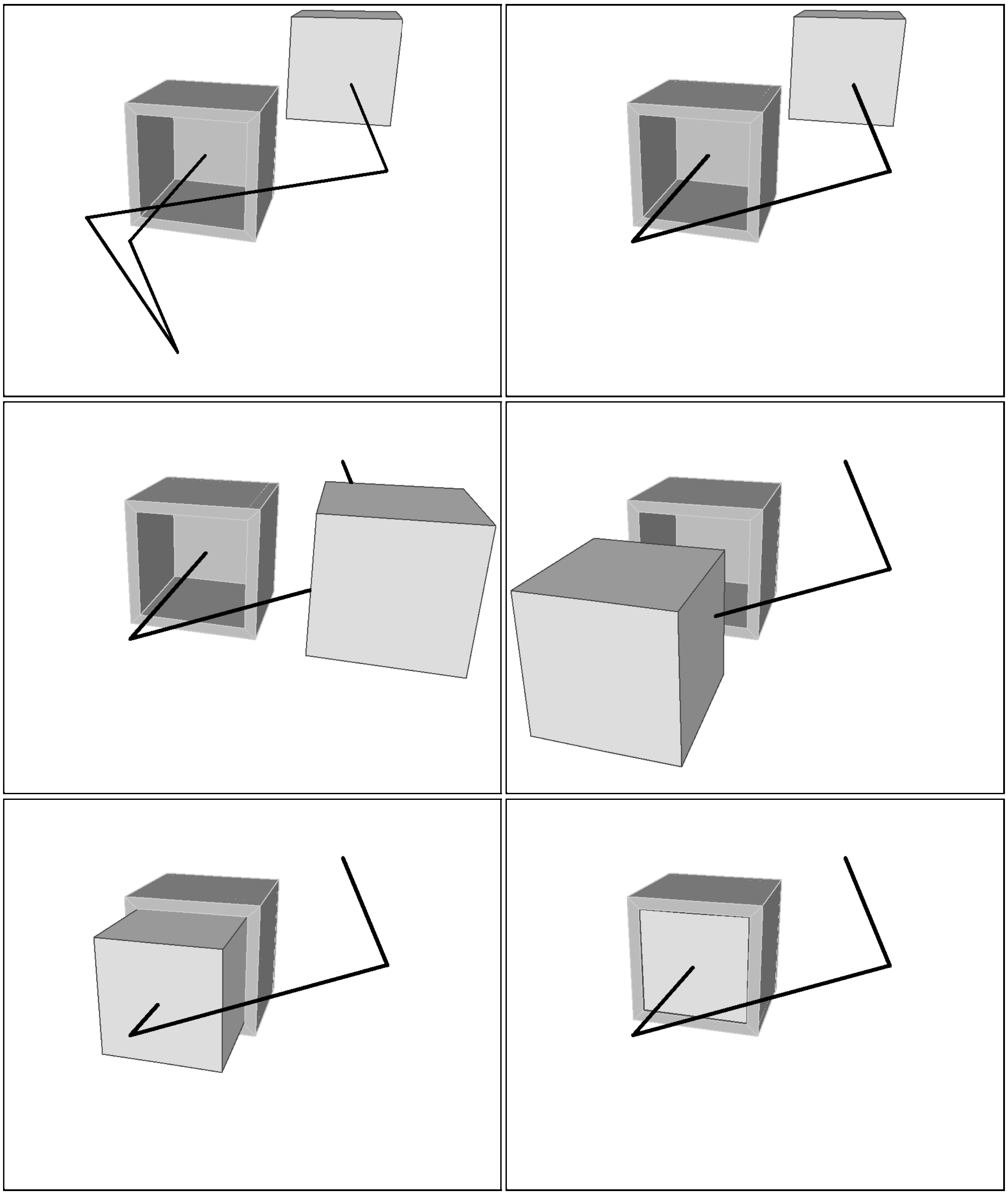


Abbildung B.1: Einfügen eines Würfels in eine Schachtel

B.2 The 'Piano Movers' Problem

In einem Zimmer befindet sich ein Piano, welches von einer Wand zu einer anderen bewegt werden soll; dabei muß es um 90° gedreht werden und einem Stuhl ausweichen. Der in dieser Szene verbleibende Freiraum ist für das hier betrachtete Bewegungsplanungsverfahren bereits sehr eng; diese Aufgabe stellt eine Herausforderung dar und benötigt zu ihrer Lösung demgemäß einige Stunden. Man beachte allerdings, daß jeweils sämtliche Umwegpunkte betrachtet wurden, wobei aber jede translatorische Grenzgerade zwei unnötige Ausweichpunkte außerhalb des Zimmers generiert. Das Verfahrens verfügt über keinerlei Information hinsichtlich der Existenz eines Zimmers, es hat lediglich Schwierigkeiten, gewisse Positionen zu erreichen und verwendet diese wegen der Rekursionstiefenbeschränkung daher nicht. Hier kann eine geschickte Beschränkung der betrachteten Umwegpunkte deutliche Verbesserungen bringen. Alternativ dazu könnte man dem Verfahren mittels eines unendlichen Hindernisses außerhalb des Zimmers das Verlassen desselben explizit verbieten; ein solches Objekt widerspricht aber der hier benutzten Definition eines Polyeders und sprengt daher den Rahmen dieser Implementierung.

In Abbildung B.2 erkennt man die berechnete Lösung, welche mit minimaler Rekursionstiefe erzeugt wurde; hierzu stand nur die schlichte Strategie zur Verfügung, die von Null an nach der kleinsten möglichen Tiefe sucht. Die einzelnen Schritte der Bewegung sind von links nach rechts und oben nach unten wiedergegeben; jedes Einzelbild enthält dabei eine Draufsicht in seinem unteren und eine Vorderansicht in seinem oberen Teil. Das Zimmer samt Einrichtung ist als Drahtgittermodell zu erkennen, der translatorische Teil des Pfades ist eingezeichnet.

Die dargestellte Lösung läßt sich nun ebenfalls verkürzen, wodurch ein erstaunlich kurzer und eleganter Pfad entsteht. Dabei wird wieder versucht, zwei Stationen des Weges direkt miteinander zu verbinden, und zwar mit einer Rotation gefolgt von einer Translation. Die Bildfolge B.2 zeigt aus einem Blickwinkel von vorne oben bezüglich der Tür eine räumliche Darstellung des Bewegungsablaufs; Decke und Vorderwände sind dabei durchsichtig gemacht.

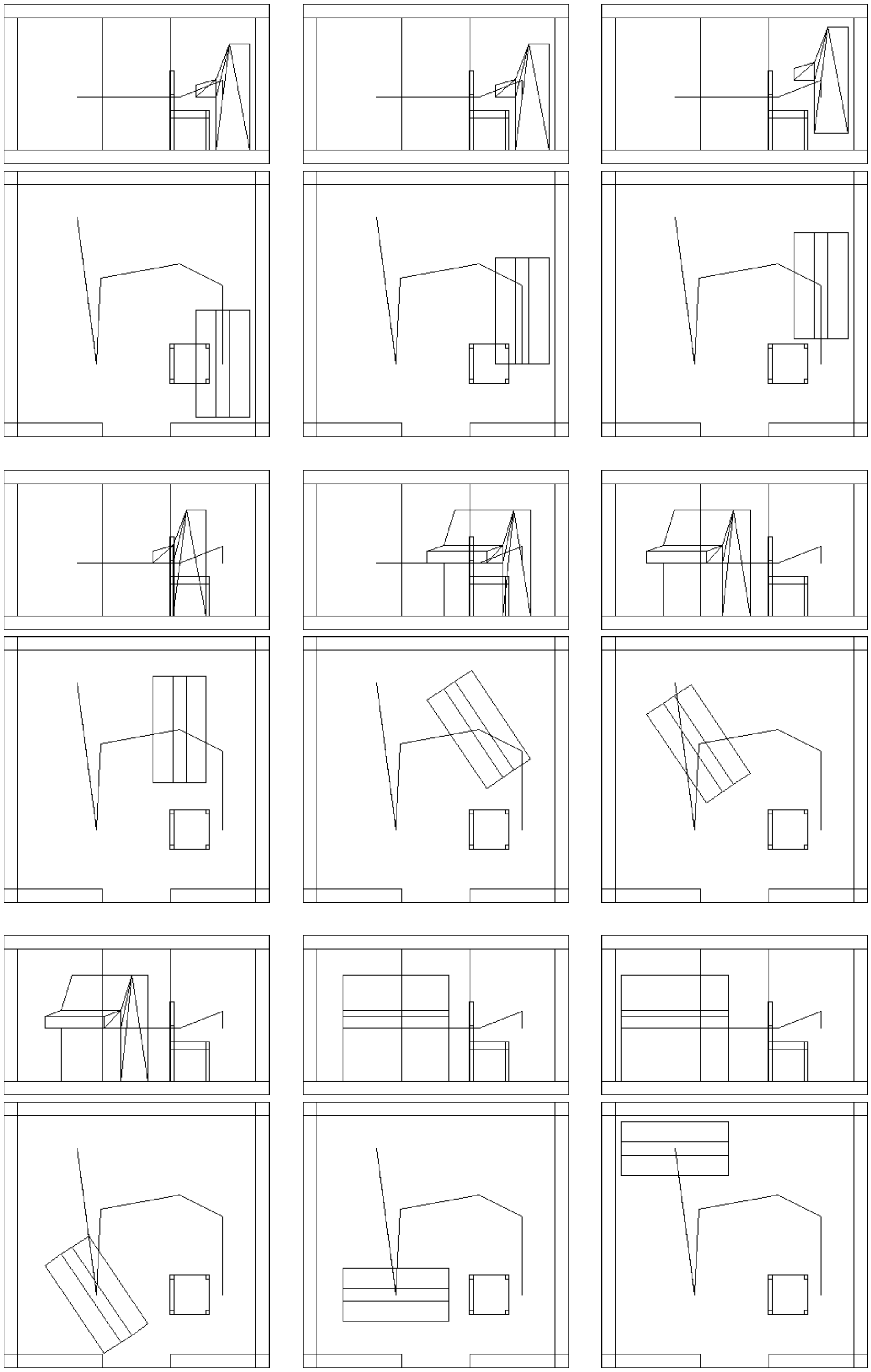
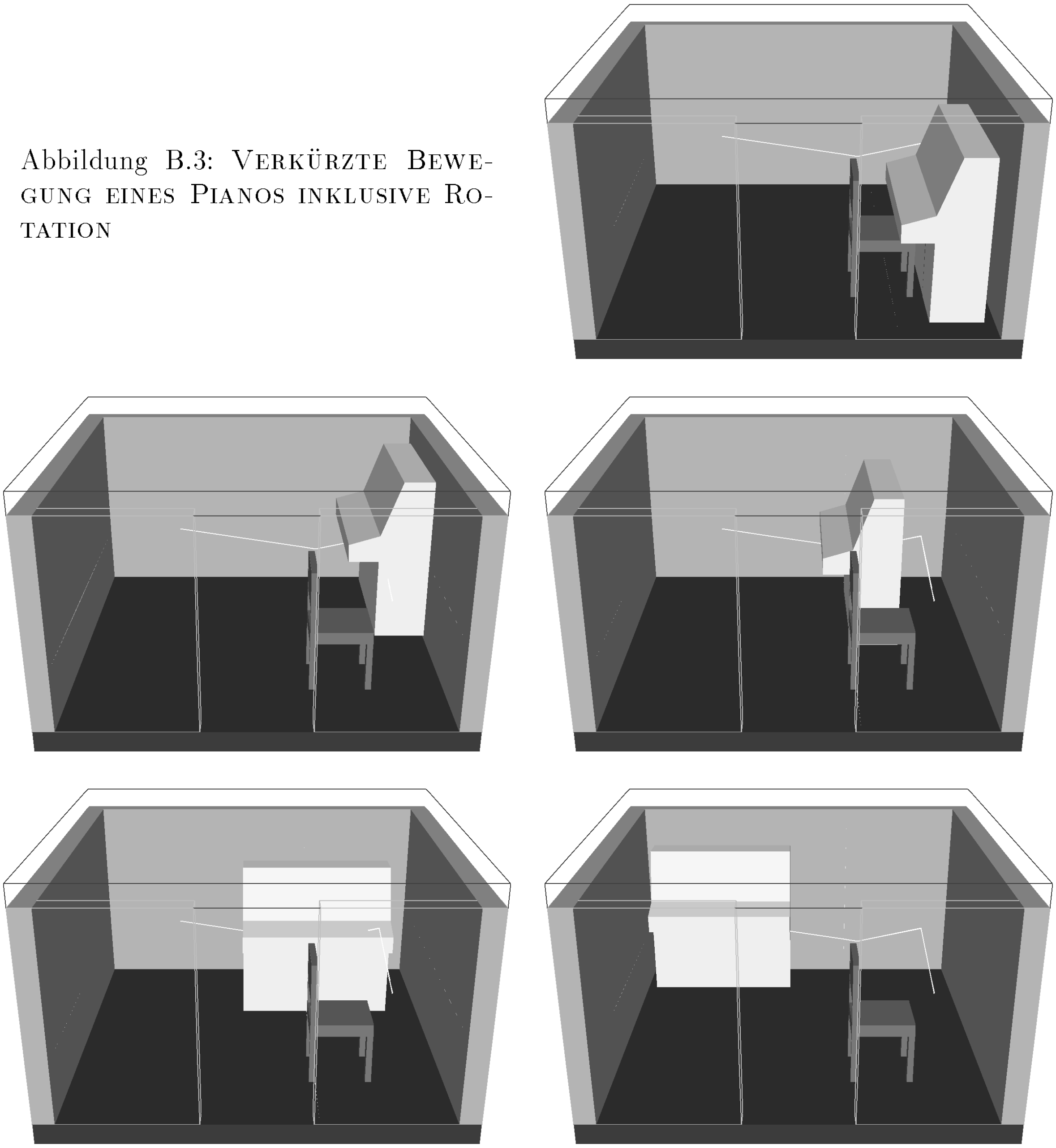


Abbildung B.2: Bewegung eines Pianos inklusive Rotation

Abbildung B.3: Verkürzte Bewegung eines Pianos inklusive Rotation



Literaturverzeichnis

[Ca87] John F. Canny
The Complexity Of Robot Motion Planning
MIT-Press, (1987)

[Col75] G. E. Collins
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition
Lecture Notes in Computer Science, No. 33, Springer-Verlag New York, (1975), S. 135–183

[La91] Jean Claude Latombe
Robot motion planning
Kluwer Academic Publishers, (1991)

[Ro87] H. Rohnert
Moving a disc between polygons
Proc. 1st Internat. Conf. Indust. Applied Math., Paris, France, (1987), S. 85–98

[Sch92] Achim Schweikard
A Simple Path Search Strategy Based on Calculation of Free Sections of Motions
Engng Applic. Artif. Intell. Vol. 5, No. 1, (1992), S. 1–10

[Sch94] Elmar Schömer
Interaktive Montageplanung mit Kollisionserkennung
Dissertation im Fachbereich Informatik, Universität des Saarlandes, (1994)

[SS82] J. Schwartz, M. Sharir
On the 'Piano Movers' Problem, II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds
Comp. Sci. Dept., New York University report 41, (1982)
auch in: „Planning, Geometry and Complexity of Robot Motion“
J. Hopcroft, J. Schwartz, M. Sharir, Ablex publishing corp. New Jersey, (1987), Kap. 5, S. 154–186