Heuristische Bewegungsplanungsstrategien im $\mathbb{R}^3$

Jens Eckstein[‡] Günter Hotz Elmar Schömer

Technischer Bericht A 05/95 März 1995

Fachbereich Informatik
Universität des Saarlandes
66123 Saarbrücken
Germany

[‡] Diplomarbeit

Abstract

Two general heuristic approaches to the geometric motion planning problem are considered. Both heuristics make use of efficient collision detection algorithms and combine motion planning with collision detection to speed up planning or to extend planning facilities.

The first approach tries to speed up motion planning by ignoring all obstacles in the scene which do not contribute to the planned collision-free path. In order to obtain such a minimal scene the heuristic computes a collision-free path in a scene S_i and checks the planned paths for collisions with ignored obstacles using standard collision detection algorithms. The colliding obstacles are included in the scene S_{i+1} and the planning process is repeated in the scene S_{i+1}. The iterative process is started in a scene S_0 without obstacles. It ends in a scene S_m which is a part of the total scene either when a collision-free path is found for the total scene or when it can be decided that there is no collision-free path in the total scene.
The heuristic thereby conserves qualitative features of the algorithms used like completeness, computation of most-secure, euclidian-shortest or time-optimal paths. It can be used with any kind of motion planning algorithms like classical ones, algorithms for dynamical environments or multiple moving objects.
The advantages of the heuristic are shown for a complete algorithm handling 3-dimensional objects with two translational degrees of freedom using the configuration space approach of Lozano-Perez and Wesley [LPW79].

The second heuristic extends the facilities of algorithms for static environments by classifying the obstacles in the scene in *fixed* and *movable* obstacles forming a motion planning problem with many degrees of freedom.
The complex problem is decomposed in a series of simple problems which can be solved efficiently by known algorithms and the solutions are combined to a solution for the whole problem.
First a collision-free path for the object to be moved is computed while ignoring the movable obstacles. In a second step the set of colliding movable obstacles is computed using standard collision detection methods. For these obstacles collision-free paths are computed separately avoiding the static and the not colliding movable obstacles so that they clear the initially computed path for the object to be moved. These paths are time-coordinated in a motion plan, so that the whole problem is solved by moving away obstacles according to the motion plan and after that moving the object along the initially computed path.
It can be shown that together with a problem modification strategy the heuristic nearly always extends and never reduces the facilities of motion planning algorithms in static environments.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Jeder Mensch ist täglich mit einer Vielzahl von Bewegungsplanungsproblemen konfrontiert, die er fast alle in Realzeit zu lösen im Stande ist. Ob man sich einen Weg durch eine Menschenmenge bahnt, den Transport eines sperrigen Gegenstandes plant oder Einzelteile zu einem neuen Gegenstand zusammenfügt, fast immer erfolgt die Lösung dieser Probleme auf einer intuitiven Ebene, so daß sich der Mensch der Komplexität der Problemstellung in der Regel nicht bewußt ist. Dabei ist er aufgrund der Flexibilität des Denkens zusätzlich in der Lage, diese Probleme mit Hilfe verschiedener Strategien zu bearbeiten, um eine für ihn „bestmögliche" Lösung zu erhalten.

Diese Fähigkeiten macht man sich bei der Lösung von Problemen dieser Art unter anderem in der Montage- und Produktionsplanung zu nutze. Hier werden die Fähigkeiten des Menschen eingesetzt, um z.B. Montagereihenfolgen festzulegen oder Bewegungsabläufe von Robotern zu entwickeln. Dabei ist eine verbreitete Methode zur Erstellung von Roboterbewegungen das *Teaching* von Robotern. Dazu nimmt der Mensch den Roboter „bei der Hand" und zeigt ihm, was er zu tun hat. Der Roboter zeichnet dies auf und kann dadurch die Bewegung erneut durchführen. Auf diese Art und Weise entstehen heutzutage Planungen für komplexe Produktionsabläufe. Dabei wird in einem langwierigen, iterativen Prozeß eine Planung erstellt, realisiert, getestet und gegebenenfalls modifiziert oder revidiert. Während dieser Zeit ist keine Produktion möglich und es entstehen verlustbringende Ausfallzeiten.

Vorteilhafter wäre es dagegen, diesen Planungsprozeß automatisch mit einer Computersimulation durchzuführen und die erhaltene Planung zu realisieren. Dadurch könnten Stillstandszeiten von Produktionsmitteln reduziert und damit die Produktivität erhöht werden.

Ein zentraler Punkt bei der Lösung derartiger Probleme stellt die Lösung verschiedenster Bewegungsplanungsprobleme dar, bei denen die Geometrie der beteiligten Objekte im Vordergrund steht. Obwohl aufgrund theoretischer Resultate bekannt ist, daß das allgemeine geometrische Bewegungsplanungsproblem lösbar ist (vgl. [SS83b]), fehlt es an praktikablen Algorithmen zur Bewegungsplanung.

Betrachtet man sich dabei die Strategien eines Menschen bei der Lösung derartiger Probleme, so bieten sich dabei Ansätze zur algorithmischen Umsetzung dieser Strategien in Bewegungsplanungsalgorithmen an. Dennoch wurde bisher der Betrachtung menschlicher Vorgehensweisen in der Entwicklung von Bewegungsplanungsalgorithmen keine hinreichende Beachtung geschenkt. Aber gerade darin liegt ein großes Potential von algorithmischen Konzepten zur Entwicklung praktikabler Bewegungsplanungsalgorithmen.

In dieser Arbeit sollen nun ausgehend von der Vorgehensweise eines menschlichen Planers algorithmische Bewegungsplanungsstrategien entwickelt werden, die die Flexibilität menschlichen Denkens mit der Fähigkeit des Computers verbinden, komplexe Probleme zu handhaben. Ein Motiv besteht dabei darin, analog zur menschlichen Vorgehensweise, komplexe Probleme in einfachere zu zerlegen, die eine schnelle Lösung erlauben und aus denen dann eine Lösung für das komplexe Problem kombiniert werden kann, wobei eventuell die Vollständigkeit der Lösung verloren geht, man jedoch dadurch eine praktikable Vorgehensweise erhält.

Gliederung und Resultate der Arbeit

In *Kapitel 1* werden zunächst einige grundlegende Begriffe der geometrischen Bewegungsplanung eingeführt.

In *Kapitel 2* wird eine allgemeine heuristische Bewegungsplanungsstrategie beschrieben, die zum Vorbild die Vorgehensweise eines menschlichen Planers hat, der eine gegebene Szene zur Lösung eines Bewegungsplanungsproblems auf die Betrachtung einer Szene zurückführt, die nur noch die zur Lösung des Problems relevanten Objekte enthält.

Dabei wird eine Folge $B_0 \subset B_1 \subset \ldots \subset B_n$ von Szenen betrachtet, auf die bekannte Bewegungsplanungsalgorithmen angesetzt werden. Die Heuristik endet, wenn entweder in einer der Teilszenen der gesamten Szene ein kollisionsfreier Weg für die gesamte Szene gefunden wird oder in der gesamten Szene kein Weg gefunden werden kann.

Zum Test, ob ein berechneter Weg in einer Teilszene kollisionsfrei in der gesamten Szene ist, werden Kollisionserkennungsalgorithmen verwendet. Dazu werden bekannte Algorithmen zur Kollisionserkennung im $\mathbb{R}^3$ beschrieben und zusätzlich effiziente Algorithmen zur Kollisionserkennung im $\mathbb{R}^2$ entwickelt.

Die Heuristik ist auf alle klassischen Bewegungsplanungsalgorithmen, Bewegungsplanungsalgorithmen in dynamischen Umgebungen und Bewegungsplanungsalgorithmen für mehrere bewegte Objekte anwendbar und erzielt zusammen mit Bewegungsplanungsalgorithmen, die auf dem Konfigurationsraumansatz basieren, die besten Ergebnisse.

Für die Heuristik werden die folgenden Eigenschaften gezeigt:

- Die Heuristik terminiert genau dann immer, wenn der benutzte Bewegungsplanungsalgorithmus für alle gestellten Bewegungsplanungsprobleme terminiert.
- Falls der benutzte Bewegungsplanungsalgorithmus für alle Eingaben terminiert, erweitert die Heuristik die Lösungsfähigkeit des Bewegungsplanungsalgorithmus höchstens.
- Ist der benutzte Bewegungsplanungsalgorithmus vollständig, so erhält man durch die Anwendung der Heuristik einen vollständigen Algorithmus.
- Berechnet der Bewegungsplanungsalgorithmus kürzeste, sicherste oder zeitminimale Wege, so berechnet die Heuristik ebenso kürzeste, sicherste oder zeitminimale Wege.

Die Heuristik versucht also bekannte Bewegungsplanungsalgorithmen zu beschleunigen, ohne deren Lösungsmöglichkeiten zu beeinträchtigen.

In *Kapitel 3* wird eine Instanz der allgemeinen Heuristik mit einem vollständigen Bewegungsplanungsalgorithmus beschrieben, der in einem 3-dimensionalen Objektraum euklidisch kürzeste translatorische Bewegungen in einer Ebene für ein bewegtes Objekt berechnet und im Rahmen dieser Arbeit implementiert wurde. Außerdem wird dessen theoretische Laufzeit analysiert. In *Anhang A* werden zu dieser Implementierung Laufzeitergebnisse bei praktischen Problemen mit und ohne Einsatz der heuristischen Bewegungsplanungsstrategie angegeben. Dabei wird deutlich, daß in größeren Szenen, in denen in der Regel bei weitem nicht alle Hindernisse zur Beschreibung der Lösung eines Bewegungsplanungsproblem beitragen, durch die heuristische Strategie enorme Laufzeitgewinne erzielt werden können, die in einem betrachteten Beispiel bis zu 94 % der Laufzeit bei klassischer Vorgehensweise betragen.

In *Kapitel 4* wird eine allgemeine Bewegungsplanungsheuristik beschrieben, die zum Vorbild die Vorgehensweise eines menschlichen Planers hat, der Hindernisse in einem Bewegungsplanungsproblem zunächst aus dem Weg räumt, um das Problem für das zu bewegende Objekt zu lösen.

Dazu werden die Hindernisse in der Szene in *feste Hindernisse* und *bewegbare Hindernisse* eingeteilt und zunächst eine Problemlösung für das zu bewegende Objekt unter Berücksichtigung der *festen Hindernisse* berechnet. Mit Hilfe der Kollisionserkennung werden aus der Menge der *bewegbaren Hindernisse* diejenigen ausgewählt, die aus dem Weg geräumt werden müssen, um die bereits berechnete Bewegung des zu bewegenden Objektes zu ermöglichen. Für diese werden sukzessive Ausweichbewegungen berechnet und anschließend koordiniert, so daß ein Bewegungsplan zur gesamten Lösung des Problems entsteht.

Zusätzlich wird eine Problemmodifikationsstrategie für die allgemeine Bewegungsplanungsstrategie beschrieben, so daß folgende Eigenschaften gezeigt werden können:

- Die Heuristik kann alle Bewegungsplanungsprobleme lösen, die mit Hilfe eines klassischen Bewegungsplanungsalgorithmus für das bewegte Objekte gelöst werden können.
- Die Heuristik erweitert die Lösungsmöglichkeiten der klassischen Bewegungsplanung.

Die konkrete Vorgehensweise der Heuristik wird in *Anhang B* an Beispielen erläutert.

Abschließend wird in *Anhang C* das IGOR-System beschrieben, im dem der in *Kapitel 3* beschriebene Algorithmus realisiert wurde.

Kapitel 1

Grundlagen der geometrischen Bewegungsplanung

In diesem Kapitel werden einige grundlegende Begriffe erklärt und Probleme der geometrischen Bewegungsplanung behandelt, auf die in den darauffolgenden Kapiteln zurückgegriffen wird.

Bei den meisten geometrischen Bewegungsplanungsalgorithmen, d.h. solchen, die nur die geometrische Beschreibungen der Objekte berücksichtigen, werden die beteiligten Körper in der Ebene als Polygone und im Raum als Polyeder modelliert. Diese weisen für die Berechnungen der Bewegungsplanung angenehme Eigenschaften wie ihre linear beschreibbare Begrenzung auf. Desweiteren kann jeder Körper mit Hilfe von Polyedern bzw. Polygonen beliebig genau approximiert werden. Daher wird in der Folge nur auf Polygone im $\mathbb{R}^2$ und Polyeder im $\mathbb{R}^3$ eingegangen.

1.1 Geometrie und Bewegungen in der Ebene

Polygone bestehen aus *Ecken* und *Kanten*, wobei Ecken Punkte im $\mathbb{R}^2$ und Kanten Konvexkombinationen zwischen zwei Punkten des $\mathbb{R}^2$ sind. In [PS85] wird ein Polygon folgendermaßen definiert:

Definition 1.1 (Polygon, einfaches Polygon)

Ein **Polygon** *ist eine endliche Menge von Liniensegmenten, deren Endpunkte in genau zwei Liniensegmenten vorkommen. Die Liniensegmente heißen* **Kanten** *und die Endpunkte* **Ecken** *des Polygons.*
Ein Polygon heißt **einfach**, *wenn sich zwei Kanten höchstens in ihren Endpunkten schneiden.*

Bezeichnung

Sei P ein Polygon. Man bezeichnet mit $int(P)$ die von P umschlossene Region ohne deren Begrenzung und mit $bound(P)$ die Begrenzung der von P umschlossenen Region.

Bemerkung 1.2 :

1. *In der Bewegungsplanung werden in der Regel nur einfache Polygone betrachtet.*
2. *Ein Polygon ist damit durch die Angabe seiner Eckpunkte im Gegenuhrzeigersinn eindeutig bestimmt.*

Ein Polygon teilt die Ebene in zwei Regionen ein, das *Innere* und das *Äußere* eines Polygons, wobei das Innere die durch das Polygon beschränkte Region und das Äußere die unbeschränkte Region ist. Soll als das Innere die unbeschränkte Region angesehen werden, so spricht man von einer *polygonalen Region.*
Eine spezielle Klasse von Polygonen stellen die konvexen Polygone dar.

Definition 1.3 (konvexes Polygon)

Ein einfaches Polygon P heißt **konvex**, *wenn für zwei beliebige Punkte $\mathbf{p}_1, \mathbf{p}_2 \in P$ gilt:*

$$\alpha \mathbf{p}_1 + (1-\alpha)\mathbf{p}_2 \in P \quad \text{für} \quad 0 \leq \alpha \leq 1$$

d.h. das Liniensegment, das beide Punkte verbindet, liegt vollständig im Inneren des Polygons.

Die Position jedes Polygons läßt sich durch die Position eines Referenzpunktes eindeutig beschreiben. Für diesen Referenzpunkt muß die Verschiebung von einem globalen Referenzpunkt (i.a. der Nullpunkt des Koordinatensystems) und die Drehung einer Referenzlinie in bezug auf eine globale Referenzlinie angegeben werden.

Zur Positionsänderung von Polygonen werden Transformationen benutzt, die Abstände erhalten und spiegelfrei sind. Diese werden als *starre Transformationen* bezeichnet. Spezielle Klassen von starren Transformationen sind Translationen und Rotationen.

Definition 1.4 (Translation, Rotation)

Sei P ein beliebiges Polygon und τ eine starre Transformation für P.
τ heißt **Translation**, *wenn τ die Orientierung der Referenzlinie von P bzgl. der globalen Referenzlinie unverändert läßt.*
τ heißt **Rotation**, *wenn durch τ die Verschiebung des Referenzpunktes von P bzgl. des globalen Referenzpunktes unverändert bleibt.*

Während sich Translationen einfach mit Hilfe eines Verschiebungsvektors beschreiben lassen, ist die Angabe von Rotationen aufwendiger.
Eine Rotation in der Ebene wird eindeutig bestimmt durch einen Rotationspunkt $\mathbf{r}$, um den die Rotation erfolgt, und einen Rotationswinkel φ, um den ein Polygon im Gegenuhrzeigersinn gedreht wird. Erfolgt die Rotation um den Nullpunkt, läßt sich ein Punkt $\mathbf{p}$ nach einer Rotation bestimmen durch

$$\mathbf{p}^{rot} = \cos\varphi\, \mathbf{p} + \sin\varphi\, \mathbf{p}^{\perp}$$

Mit

$$\mathbf{p}^{\perp} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{p}$$

ergibt sich die zugehörige Rotationsmatrix zu

$$\mathbf{R} = \cos\varphi\,\mathbf{I} + \sin\varphi \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Die Menge aller so definierten Rotationsmatrizen bilden die zyklische Gruppe $SO(2)$.

Die Rotation eines Punktes $\mathbf{p}$ um einen beliebigen Rotationspunkt $\mathbf{r}$ wird dann berechnet durch

$$\mathbf{p}^{rot} = \mathbf{R}(\mathbf{p} - \mathbf{r}) + \mathbf{r}$$

Bemerkung 1.5 :

Sei P ein beliebiges Polygon. Jede Positionsveränderung von P läßt sich als Hintereinanderausführung einer Rotation und einer Translation beschreiben. D.h. die Position eines Polygons läßt sich durch die Angabe eines Punktes in dem Raum $\mathbb{R}^2 \times SO(2)$ beschreiben.

Von Bewegungsplanungsalgorithmen werden Wege für Polygone berechnet, die stetige Positionsänderungen beinhalten. Diese werden wie folgt definiert:

Definition 1.6 (Weg)
Sei P ein beliebiges Polygon und p_{start} und p_{end} zwei beliebige Positionen für P. Eine stetige Abbildung

$$\begin{aligned} \pi : [\,0,1] &\to \mathbb{R}^2 \times SO(2) \\ t &\mapsto \pi(t) \end{aligned} \qquad \textit{mit} \quad \pi(0) = p_{start}\ ,\ \pi(1) = p_{end}$$

heißt **Weg** *für P von p_{start} nach p_{end}.*

1.2 Geometrie und Bewegungen im Raum

Polyeder bestehen aus **Ecken**, die Punkte im $\mathbb{R}^3$ sind, **Kanten**, die Konvexkombinationen zwischen Ecken sind, und **Flächen**, Polygonen, eingebettet in den $\mathbb{R}^3$. In [PS85] wird ein Polyeder wie folgt definiert:

Definition 1.7 (Polyeder, einfaches Polyeder)
Ein **Polyeder** *im $\mathbb{R}^3$ ist eine Menge von ebenen Polygonen, von denen jeweils genau zwei Polygone eine Kante gemeinsam haben. Solche Polygone heißen adjazent.*
Die Ecken der Polygone sind die Ecken des Polyeders, die Kanten der Polygone sind die Kanten des Polyeders und die Polygone sind die Flächen des Polyeders.
Ein Polyeder heißt **einfach**, *wenn es kein Paar von nicht-adjazenten Flächen gibt, die einen Punkt gemeinsam haben.*

Die Begriffe $int(P)$ und $bound(P)$ werden analog zu Polygonen verwendet.

Die von den Flächen des Polyeders beschränkte Region heißt *Inneres* und die unbeschränkte Region *Äußeres* des Polyeders. Für jede Fläche wird der Normalenvektor der von der Fläche unterstützten Ebene in das Polyederäußere gerichtet. Eine spezielle Klasse von Polyedern sind die konvexen Polyeder.

Definition 1.8 (konvexes Polyeder)
Ein Polyeder P heißt **konvex**, *wenn sein Inneres eine konvexe Menge ist, d.h. für zwei beliebige Punkte $\mathbf{p}_1$ und $\mathbf{p}_2 \in P$ gilt:*

$$\alpha\mathbf{p}_1 + (1-\alpha)\mathbf{p}_2 \in P \quad \textit{für} \quad 0 \leq \alpha \leq 1$$

Die Position eines Polyeders ist eindeutig beschrieben durch die Verschiebung eines Referenzpunktes $\mathbf{p}$ von einem globalen Referenzpunkt und die Orientierung des Polyeders. Die Orientierung eines Polyeders beschreibt die Rotation, mit Hilfe derer das Polyeder aus seiner Referenzorientierung in die aktuelle Orientierung überführt werden kann.

Während sich Translationen analog zur Geometrie in der Ebene durch Verschiebungsvektoren beschreiben lassen, ist die Angabe von Rotationen im Raum gegenüber der Ebene modifiziert. Eine Rotation im Raum wird eindeutig bestimmt durch eine Rotationsachse $\mathbf{r}$ und einen Rotationswinkel φ. Verläuft die Rotationsachse durch den Nullpunkt, läßt sich für jeden Punkt $\mathbf{p}$ der rotierte Punkt $\mathbf{p}^{rot}$ bestimmen durch

$$\mathbf{p}^{rot} = (\mathbf{r}^T\mathbf{p})\mathbf{r} + \cos\varphi\,\mathbf{r} \times (\mathbf{p} \times \mathbf{r}) + \sin\varphi\,\mathbf{r} \times \mathbf{p}$$

Aus dieser Gleichung kann folgende Rotationsmatrix $\mathbf{R}$ abgeleitet werden, mit der $\mathbf{p}$ von links multipliziert werden muß, um $\mathbf{p}^{rot}$ zu erhalten:

$$\mathbf{R} = (1 - \cos\ \varphi)\mathbf{r}\mathbf{r}^T + \cos\ \varphi\,\mathbf{I} + \sin\ \varphi \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$

Für Rotationen um beliebige Achsen läßt sich der rotierte Punkt $\mathbf{p}^{rot}$ berechnen durch

$$\mathbf{p}^{rot} = \mathbf{R}(\mathbf{p} - \mathbf{p_r}) + \mathbf{p_r} \quad \text{mit} \quad \mathbf{p_r} \text{ beliebiger Punkt der Rotationsachse}$$

Die Menge der so definierten Rotationsmatrizen bilden die zyklische Gruppe $SO(3)$. Somit läßt sich die Position eines Polyeders durch die Angabe eines Punktes im Raum $\mathbb{R}^3 \times SO(3)$ beschreiben. Analog zur Geometrie in der Ebene lassen sich auch im Raum alle starren Transformationen von Polyedern als eine Folge von einer Rotation und einer Translation darstellen. Der Begriff des Weges wird im Raum analog zur Ebene definiert.

1.3 Bewegte Objekte und Hindernisse in der Bewegungsplanung

Nach dieser grundlegenden Einführung in die Geometrie und die Bewegungen von Polygonen im $\mathbb{R}^2$ bzw. Polyedern im $\mathbb{R}^3$ muß an dieser Stelle noch auf ein grundsätzliches Problem der Bewegungsplanung eingegangen werden:

Die Modellierung von bewegten Objekten und Hindernissen als offene oder abgeschlossene Mengen

Grundsätzlich geht man bei den Bewegungsplanungsalgorithmen davon aus, daß die zu bewegenden Objekte abgeschlossene Mengen sind, während die Hindernisse als offene Mengen definiert werden. Dies wird z.B. in [LPW79], [SS83a] oder [SS83b] so gehandhabt. Die Definition zumindest der Hindernisse als offene Objekte hat zur Folge, daß eine Bewegung π eines Objektes O als kollisionsfrei bzgl. eines Hindernisses H angesehen wird, wenn während der gesamten Bewegung der Schnitt zwischen dem bewegten Objekt und dem Inneren des Hindernisses leer ist, d.h.

$$\pi \text{ kollisionsfrei für } O \text{ bzgl. } H \iff O(\pi(t)) \cap int(H) = \emptyset \text{ für } t \in [0, 1]$$

Es ist offensichtlich, daß dann auch für den Fall, daß alle Objekte in einer Szene als offen angesehen werden, die Bewegung für das bewegte Objekt O kollisionsfrei ist bzgl. eines Hindernisses H.

In der praktischen Implementierung im Rahmen des IGOR-Systems hat sich jedoch ergeben, daß die unterschiedliche Behandlung der Objekte in einer Szene während der Kollisionserkennung – alle Objekte als abgeschlossene Mengen – und der Bewegungsplanung – bewegtes Objekt als abgeschlossene Menge, Hindernisse als offene Mengen – zu keinen Problemen bei der Verifikation der Kollisisionsfreiheit der geplanten Bewegungen geführt hat. Deshalb werden bei der Beschreibung der Kollisionserkennung in *Kapitel 2* alle Objekte als abgeschlossene Mengen und bei der Beschreibung des Bewegungsplanungsalgorithmus in *Kapitel 3* das bewegte Objekt als abgeschlossene Menge und die Hindernisse als offene Mengen behandelt.

Konfigurationsräume

Definition 1.9 (Konfigurationsraum)
Sei O ein beliebiges Objekt im $\mathbb{R}^n$. Dann heißt der Raum $\mathbb{R}^n \times SO(n)$ aller Positionen von O **Konfigurationsraum** *von O, wobei $SO(n)$ die zyklische Gruppe der Rotationsmatrizen im $\mathbb{R}^n$ darstellt.*

In [Ud77] und [LPW79] wurde der Konfigurationsraumansatz eingeführt, in dem das spezifizierte Bewegungsplanungsproblem im Objektraum – dem Raum, in dem das Objekt bewegt wird – transformiert wird in den zugehörigen Konfigurationsraum, in dem jeder Punkt eindeutig einer Position des bewegten Objektes im Objektraum entspricht.

Die Punkte des Konfigurationsraumes können nun danach klassifiziert werden, ob das bewegte Objekt in dieser Position kein Hindernis schneidet oder berührt, ein Hindernis schneidet oder ein Hindernis berührt.

Definition 1.10 (Freie, halbfreie, unfreie Positionen)
Sei O ein beliebiges Objekt und H eine Menge von beliebigen Hindernisobjekten im Objektraum. Eine Position p von O heißt

1. **frei**, *falls* $O(p) \cap \bigcup_{h \in H} h = \emptyset$,
2. **unfrei**, *falls* $O(p) \cap \bigcup_{h \in H} int(h) \neq \emptyset$,
3. **halbfrei**, *falls* $O(p) \cap \bigcup_{h \in H} int(h) = \emptyset$ *und* $O(p) \cap \bigcup_{h \in H} bound(h) \neq \emptyset$.

Die freien Positionen entsprechenden Punkte des Konfigurationsraumes heißen *Freiraum* und werden durch halbfreie Positionen von den unfreien Positionen getrennt. Daher eignen sich die halbfreien Positionen zur Beschreibung des Freiraums.

Definition 1.11 (Freiraum)
Sei O ein beliebiges Objekt und H eine Menge von beliebigen Hindernisobjekten im Objektraum. Die Menge aller freien Positionen von O bzgl. H wird als **Freiraum** *FP des Konfigurationsraumes P bezeichnet.*

$$FP = \left\{ p \in P \mid O(p) \cap \bigcup_{h \in H} h = \emptyset \right\}$$

Die Hindernisse im Objektraum werden durch die Bildung des Konfigurationsraumes abgebildet auf zugehörige Konfigurationsraumhindernisse, deren Aussehen zusätzlich von dem bewegten Objekt O abhängt. D.h. die Bildung der Konfigurationsraumhindernisse ist eine Abbildung

$$\begin{gathered} \pi_O : H \to 2^P \\ h \mapsto CO_h^O \subset P \end{gathered}$$

wobei H eine Menge von Hindernisobjekten und P die Menge aller möglichen Positionen des bewegten Objektes ist.

Damit läßt sich der Freiraum im Konfigurationsraum beschreiben als das Komplement der Vereinigung aller Konfigurationsraumhindernisse; es gilt:

$$FP = P \cap \overline{\bigcup_{h \in H} CO_h^O}$$

Dieser Freiraum kann in Zusammenhangskomponenten zerfallen, wobei eine Zusammenhangskomponente definiert ist als

Definition 1.12 (Zusammenhangskomponente des Freiraums)
Sie FP der Freiraum eines beliebigen Konfigurationsraumes P und $\mathbf{p}$ *ein Punkt des Freiraumes. Die* **Zusammenhangskomponente** *von FP, die* $\mathbf{p}$ *enthält, ist die größte Teilmenge ZHK von FP, die* $\mathbf{p}$ *enthält und in der alle Punkte* $\mathbf{q}$ *über einen Weg in FP von* $\mathbf{p}$ *aus erreichbar sind.*

$$ZHK_{\mathbf{p}}^{FP} = \{\mathbf{q} \in FP \mid \exists\, \pi : [\,0,1] \rightarrow FP \;\; \pi \textit{ stetig: } \pi(0) = \mathbf{p}, \pi(1) = \mathbf{q}\}$$

Da kollisionsfreie Bewegungen für Objekte über freie und halbfreie Positionen verlaufen, entspricht eine Zusammenhangskomponente des Freiraums, die die aktuelle Position eines Objektes enthält, zusammen mit ihrer Randbeschreibung der Menge aller von dieser Position aus kollisionsfrei erreichbaren Positionen des Objektes.

Kapitel 2

Heuristisch inkrementelle Bewegungsplanung im $\mathbb{R}^3$

Die Ende der siebziger und Anfang der achtziger Jahre entwickelten Algorithmen zur Bewegungsplanung versuchten, vollständige Lösungen für das Bewegungsplanungsproblem zu finden. Dabei wurde das Bewegungsplanungsproblem für die Objekte im Objektraum transformiert auf ein Bewegungsplanungsproblem für einen Punkt im Raum aller möglichen Positionen des zu bewegenden Objektes (Konfigurationsraum). Diese Vorgehensweise wurde von [Ud77] und [LPW79] angeregt und führte zu exakten Algorithmen für die Bewegungsplanung.

Schwartz und Sharir zeigten zudem in [SS83b], daß mit Hilfe des Konfigurationsraumansatzes das allgemeine geometrische Bewegungsplanungsproblem lösbar ist, falls bewegte Objekte und Hindernisse als semialgebraische Mengen beschrieben sind. Die Menge aller freien Plazierungen des zu bewegenden Objektes, d.h. der Freiraum innerhalb des Konfigurationsraumes, läßt sich dann ebenfalls als semialgebraische Menge darstellen und mit Hilfe des Algorithmus von Collins (vgl. [Col75]) kann eine Zerlegung des Freiraums in Zellen berechnet werden. Die Zellen werden zu den Knoten eines Graphen, in dem Zellen durch Kanten miteinander verbunden werden, wenn es eine kollisionsfreie Bewegung für das zu bewegende Objekt zwischen den beiden Zellen gibt, ohne eine andere Zelle des Freiraums zu benutzen. Damit reduziert sich das gesamte Bewegungsplanungsproblem auf ein Suchproblem in einem Graphen. Allerdings hat der Algorithmus doppelt exponentielle Laufzeit in der Anzahl der Freiheitsgrade des Problems. Canny erreichte in [Ca87] eine Verbesserung auf einfach exponentielle Laufzeit, jedoch hat die Vorgehensweise für praktische Fragestellungen keine Bedeutung.

Die Entwicklungen in der algorithmischen Bewegungsplanung gingen und gehen dahin, daß einerseits versucht wird, die Berechnung von Zusammenhangskomponenten des Freiraums zu beschleunigen (vgl. [GSS89] und [CEG+93]) und andererseits mit Hilfe von Heuristiken die Berechnung des kompletten Konfigurationsraums zu umgehen, indem wie z.B. in [Sch92a] nur lokale Ausschnitte des Konfigurationsraumes bei Bedarf berechnet werden. Gleichzeitig wird auch versucht, die Komplexität von Bewegungsplanungsproblemen anstatt klassisch über die Anzahl und Komplexität der Beschreibung der in einem Problem gegebenen Objekte durch alternative Komplexitätsmaße zu beschreiben (vgl. [Sch92]), indem die intuitive Schwierigkeit eines Bewegungsplanungsproblems in ein Komplexitätsmaß gefaßt wird und die Laufzeit des Algorithmus zusätzlich von diesem neuem Komplexitätsmaß abhängt.

2.1 Motivation

Wie sich allerdings in der praktischen Realisierung eines Bewegungsplanungssystems im 3-dimensionalen Objektraum mit zwei translatorischen Freiheitsgraden und der Berechnung kürzester Wege zeigt, wird der größte Teil der Gesamtlaufzeit zur Berechnung der Lösung entgegen der theoretischen Analyse nicht in der Berechnung des Sichtbarkeitsgraphen, sondern zur Berechnung des Konfigurationsraumes bzw. zur Berechnung der Zusammenhangskomponente des Freiraumes im Konfigurationsraum, die den Startpunkt der Bewegung enthält, verbraucht (vgl. [ES93]). Die Analyse der Aufteilung der Gesamtlaufzeit zur Problemlösung auf die einzelnen zu lösenden Teilprobleme zeigt, wie aus Abbildung 2.1 ersichtlich, daß in der Berechnung des Konfigurationsraumes das größte Zeiteinsparungspotential liegt.

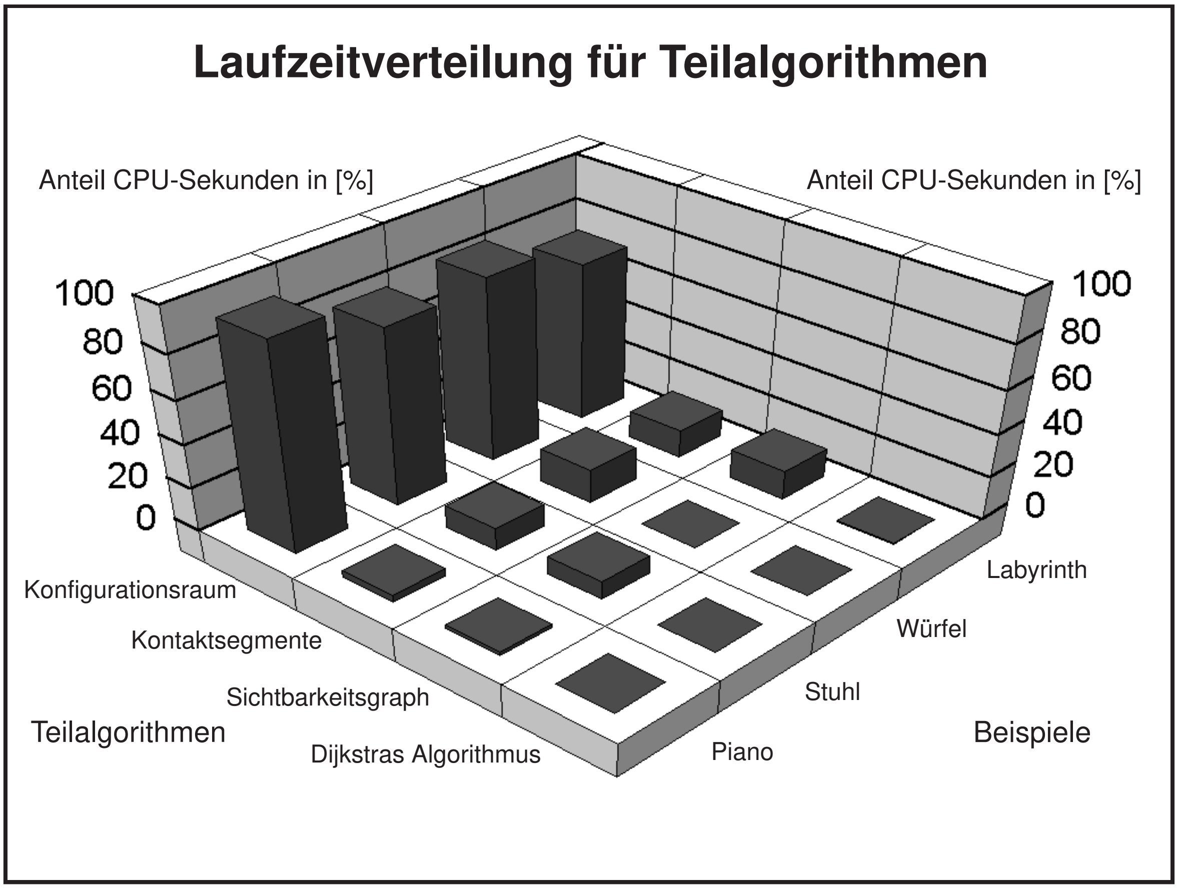

Abbildung 2.1: Gemessene Laufzeitverteilungen für Bewegungsplanungsprobleme

Wenn man die praktischen Ergebnisse betrachtet, die sich aus [ES93] ergeben, und insbesondere die dabei berechneten Konfigurationsräume und Wege, so fällt auf, daß bei komplexer werdenden Szenen die Lösung der Probleme i.a. nicht von allen in der Szene befindlichen Hindernissen abhängt. Anstatt dessen kann man zur Lösung des gestellten Bewegungsplanungsproblems eine Szene betrachten, die nur eine Teilmenge der Hindernisse enthält, und in dieser eine Lösung berechnen, die gleichzeitig eine Lösung für das gestellte Bewegungsplanungsproblem ist. Daraus resultiert eine vereinfachte Problemstellung und damit eine schnellere Lösung des Bewegungsplanungsproblems. Dabei ist die Menge der zur Lösung des Bewegungsplanungsproblems benötigten Hindernisse abhängig von dem eingesetzten Bewegungsplanungsalgorithmus.

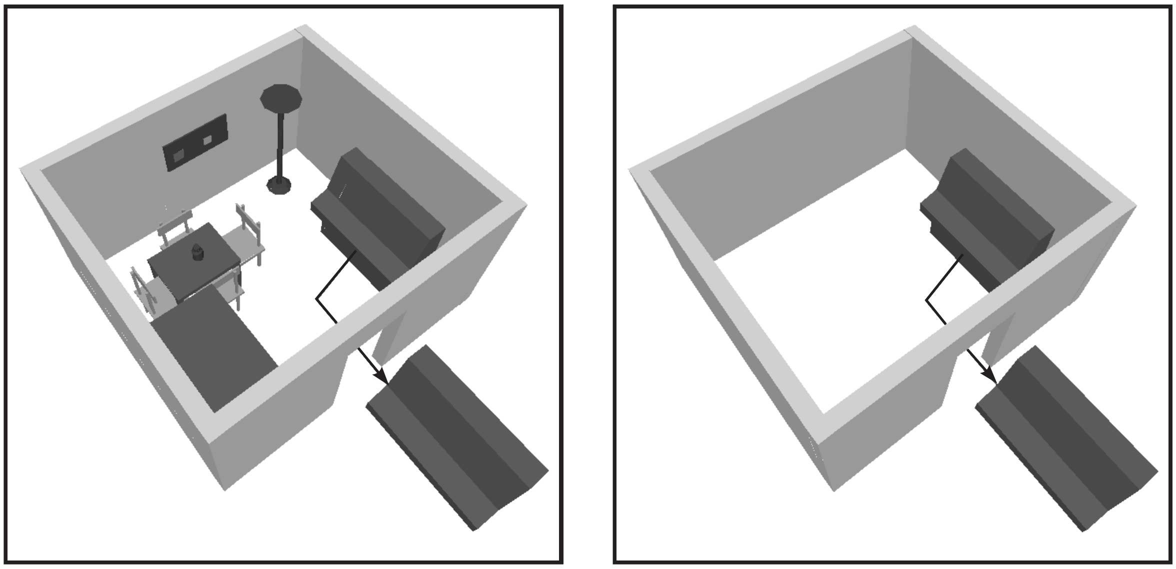

Abbildung 2.2: Zwei Bewegungsplanungsprobleme mit identischer Lösung bei kürzesten Wegen

Wie man in Abbildung 2.2 sehen kann, führen die beiden Bewegungsplanungsprobleme zu identischen Lösungen, allerdings ist im zweiten Fall nur eine Teilmenge von Hindernissen im Objektraum vorhanden, so daß die Berechnung der Lösung dieses Problemes deutlich schneller erfolgt als die des ersten Problems. Gleichzeitig ist aber die Lösung des zweiten Problems auch eine Lösung des ersten Problems, d.h. durch geeignete Vernachlässigung von Hindernissen im Objektraum läßt sich die Berechnung einer Lösung beschleunigen.

2.2 Beschreibung der heuristisch inkrementellen Vorgehensweise

Wie man bei Betrachtung von praktischen Beispielen sieht, ist es i.a. möglich, die Szene vor Berechnung der Lösung eines Bewegungsplanungsproblems so zu vereinfachen, daß durch die Lösung des modifizierten Problems eine Lösung für das gestellte Problem gefunden wird, die Berechnung der Lösung jedoch vereinfacht wird. In der Folge soll nun eine Strategie zur Berechnung dieser vereinfachten Szene entwickelt werden.

Als Vorbild für die Strategie dient die Vorgehensweise eines „menschlichen Planers“ bei der Lösung eines Bewegungsplanungsproblems. Ein menschlicher Planer analysiert bei einem Bewegungsplanungsproblem zunächst die grobe Struktur der Hinderniswelt und sucht dabei nach einem ungefähren Verlauf eines möglichen Lösungsweges. Hat er diesen ungefähren Verlauf gefunden, so versucht der Planer den Weg genauer zu spezifizieren und zu verifizieren, d.h. er bezieht Hindernisse, die in der näheren Umgebung seines bisher grob geplanten Weges stehen, in die weitere Planung ein und ignoriert weiter entfernt stehende, von denen er annimmt, daß sie für die weitere Wegeplanung keine Bedeutung haben. Er reduziert also die von ihm betrachtete Szene auf einen Korridor um den von ihm in einer ersten groben Annäherung geplanten Weg. Entlang dieses Weges betrachtet er die Szene detaillierter und modelliert seinen Weg genauer. Ist der dadurch entstandene Wegeplan für den Planer sicher als kollisionsfrei anzusehen, so beendet er die Planung. Gibt es allerdings noch Stellen in dem Wegeplan, an denen er sich nicht sicher ist, daß das Objekt an diesen kollisionsfrei bewegt werden kann, so verifiziert er diese noch genauer. Innerhalb dieses iterativen Prozesses kann es natürlich vorkommen, daß sich die ursprüngliche Grobplanung als fehlerhaft erweist und der Planungsvorgang erneut begonnen werden muß.

Beispielhaft läßt sich diese Vorgehensweise an dem Transport eines Pianos in ein Haus verdeutlichen:

> In der ersten Stufe der Planung des Weges für das Piano wird der Planer nicht die Kaffeetasse auf dem Küchentisch berücksichtigen, obwohl ihm die Szenerie detailliert bekannt ist. Er wird zunächst die Abfolge der Zimmer planen, durch die das Piano bewegt werden soll. Schon an dieser Stelle verliert das genaue Aussehen der nicht betroffenen Räume des Hauses jegliche Bedeutung für die weitere Wegeplanung. Danach wird er bestimmen, wie das Piano durch die verschiedenen Zimmer befördert werden soll. Dazu wird er diese Räume mit ihrer Einrichtung genauer betrachten und die Einzelheiten des Wegeplanes festlegen. Entstehen bei dieser Detailplanung Engstellen, wird er an diesen verifizieren, ob sein Bewegungsplan korrekt ist (z.B. durch Nachmessen der Durchgangsbreite). Stellt er in der Detailplanung einen Fehler in der Grobplanung fest (z.B., daß eine Durchgangsbreite nicht ausreichend ist), beginnt er die Grobplanung erneut.

Während einer solchen Wegeplanung wird trotz einer komplexen Gesamtszene, bei der a priori nicht bestimmt werden kann, durch welchen Teil der Szene der geplante Weg verlaufen wird, nur ein kleiner Teil der Hindernisse für die Planung des Weges betrachtet.

Man versucht eine möglichst einfache Bewegung zu beschreiben. Dazu entfernt man in einem ersten Schritt alle Hindernisse aus der Szene und erhält als Lösung des Bewegungsplanungsproblems die direkte Verbindung von Start- und Endposition des zu bewegenden Objektes. Den so erhaltenen Weg testet man mit allen in der ursprünglichen Szene enthaltenen Hindernisse auf Kollisionsfreiheit. Ist der berechnete Weg kollisionsfrei, ist das Problem gelöst. Treten Kollisionen auf, werden alle Hindernisse, mit denen das zu bewegende Objekt kollidiert, in die Szene zur weiteren Betrachtung aufgenommen. In der modifizierten Szene wird eine erneute Bewegungsplanung durchgeführt und der gefundene Weg auf Kollisionsfreiheit geprüft. Dies wird solange fortgesetzt, bis entweder ein kollisionsfreier Weg gefunden wird oder entschieden werden kann, daß es keinen kollisionsfreien Weg gibt.

Diese Vorgehensweise läßt sich folgendermaßen zusammenfassen:

H	Menge aller Hindernisse
H_{nicht}	Menge der aktuell **nicht** betrachteten Hindernisse
$H_{aktuell}$	Menge der aktuell betrachteten Hindernisse
O	bewegtes Objekt
p_{start}	Startposition von O
p_{end}	Zielposition on O
w	berechneter kollisionsfreier Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}

$H_{nicht} \leftarrow H$
$H_{aktuell} \leftarrow \emptyset$
$w \leftarrow$ beliebiger Weg von p_{start} nach p_{end}
while ((w existiert) und (w nicht kollisionsfrei bzgl. H_{nicht})) **do**
    forall $h \in H_{nicht}$ **do**
        if (O kollidiert mit h auf w) **then**
            $H_{aktuell} \leftarrow H_{aktuell} \cup h$
            $H_{nicht} \leftarrow H_{nicht} \setminus \{h\}$
    if ($\exists$ kollsionsfreien Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}) **then**
        $w \leftarrow$ kollisionsfreier Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}
    else
        $w \leftarrow$ existiert nicht

Damit wird das Konzept für eine allgemeine iterative Bewegungsplanungsstrategie beschrieben.

Zur Durchführung der Bewegungsplanungsstrategie muß nun in der Folge die Lösung der beiden Teilprobleme

1. Berechnung der Kollisionen zwischen bewegtem Objekt und Hindernissen

2. Berechnung von kollisionsfreien Wegen für das bewegte Objekt

betrachtet werden.

2.3 Kollisionen bei Bewegungen

Das Problem, die Kollisionsfreiheit einer gegebenen Bewegung zu zeigen oder die Menge der Hindernispolyeder zu berechnen, mit denen ein bewegtes Polyeder während einer Bewegung kollidiert, läßt sich formal beschreiben als:

gegeben:

H	Menge von Hindernispolyedern
P	bewegtes Polyeder
w	Bewegung für O

gesucht:

$H_{col} = \{h \in H \mid \exists\, t \in [0,1]: \ P(w(t)) \cap h \neq \emptyset\}$

Die Komplexität des Problems hängt von der Beschreibungskomplexität der Körper und der gegebenen Bewegung w für P ab. Da viele bekannte Bewegungsplanungsalgorithmen stückweise translatorische bzw. rotatorische Bewegungen berechnen, wird die Betrachtung in der Folge auf Translationen und Rotationen beschränkt. Für Bewegungsplanungsalgorithmen, die allgemeinere Bewegungen berechnen, muß auf andere Verfahren der Kollisionserkennung zurückgegriffen werden, z.B. numerische Verfahren, wie sie in [Sch94] und [Si94] beschrieben sind.

2.3.1 Translationen

Bei rein translatorischen Bewegungen läßt sich die oben angegebene allgemeine Fragestellung für die Kollisionserkennung auf den Fall reduzieren, daß die Bewegung w eine Translation entlang eines Vektors in Richtung $\mathbf{s}$ der Länge $|s|$ ist. Dafür wird zunächst die kanonische Vorgehensweise beschrieben und danach auf theoretisch effizientere Algorithmen eingegangen.

Formal ist folgendes Problem zu lösen:

gegeben:

P_1	bewegtes Polyeder
P_2	festes Polyeder
$\mathbf{s}$	Translationsvektor für P_1

gesucht:

$\exists\, t \in [\,0,1],\ P_1(t\mathbf{s}) \cap P_2 \neq \emptyset?$

2.3.1.1 Elementare Kollisionserkennung

Grundlage des Algorithmus ist die Beobachtung, daß, wenn zwei Polyeder, von denen einer bewegt wird, kollidieren, diese Kollision zwischen einer Ecke des einen und einer Fläche des anderen Polyeders oder zwischen zwei Kanten der Polyeder auftritt (zum Beweis vgl. Abschnitt 3.1.1, Seite 48). Dabei bedeutet eine Kollision zwischen zwei Polyedern, daß deren Begrenzungen in miteinander in Kontakt kommen. Daher kann die Kollisionserkennung zwischen zwei Polyedern auf die Betrachtung dieser beiden Fälle reduziert werden.

Ecke–Fläche–Kollision

Formal ist folgendes Problem zu lösen.

gegeben:

$\mathbf{p} \in \mathbb{R}^3$	Ecke eines Polyeders
f	Konvexe Fläche eines Polyeders dargestellt als Polygon im $\mathbb{R}^3$
$\mathbf{s} \in \mathbb{R}^3$	Translationsvektor für $\mathbf{p}$

gesucht:

$\exists\, t \in [\,0,1]$ mit $\mathbf{p} + t\mathbf{s} \in f$?

Die Fläche f sei gegeben durch eine zyklische Liste $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1}, \mathbf{v}_k \equiv \mathbf{v}_0$ ihrer Eckpunkte im Gegenuhrzeigersinn, wenn man entgegen der Richtung des Normalenvektors der unterstützten Ebene E auf die Fläche sieht. Die Ebene $E = \{\mathbf{x} \mid \mathbf{n}^T\mathbf{x} = d\}$, in der f liegt, läßt sich bestimmen durch

$$\begin{aligned} \mathbf{n} &= (\mathbf{v_2} - \mathbf{v_1}) \times (\mathbf{v_3} - \mathbf{v_1}) \neq \mathbf{0} \\ d &= \mathbf{n}^T\mathbf{v_1} \end{aligned}$$

Jeweils zwei benachbarte Eckpunkte des Polygons beschreiben einen Richtungsvektor $\mathbf{v_{i+1}} - \mathbf{v_i}$, der zusammen mit dem Normalenvektor $\mathbf{n}$ der Ebene E und einem der beiden Eckpunkten eine Ebene beschreibt, mit Hilfe derer man entscheiden kann, ob ein Punkt $\mathbf{q}$ der Ebene E innerhalb des Polygons f liegt durch

$$\begin{aligned} & \forall\, i \in \{0, \ldots, k-1\} \quad (\mathbf{n} \times (\mathbf{v_{i+1}} - \mathbf{v_i}))^T (\mathbf{q} - \mathbf{v_i}) \geq 0 \\ \Longleftrightarrow \quad & \forall i \in \{0, \ldots, k-1\} \quad \det[\mathbf{n}, (\mathbf{v_{i+1}} - \mathbf{v_i}), (\mathbf{q} - \mathbf{v}_i)] \geq 0 \end{aligned}$$

Es wird nun in zwei Schritten entschieden, ob $\mathbf{p}$ durch eine Translation $\mathbf{s}$ das Polygon f schneidet. Im ersten Schritt wird berechnet, ob $\mathbf{p}$ durch die Translation $\mathbf{s}$ die durch f unterstützte Ebene E schneidet. Dies erfolgt durch Lösung der Gleichung

$$\mathbf{n}^T(\mathbf{p} + t\mathbf{s}) = d$$

Es ergibt sich als Lösung

$$t = \frac{d - \mathbf{n}^T\mathbf{p}}{\mathbf{n}^T\mathbf{s}} \qquad \text{für } \mathbf{n}^T\mathbf{s} \neq \mathbf{0}$$

In einem zweiten Schritt wird getestet, ob der Schnittpunkt innerhalb von f liegt, falls er existiert. Sei $\mathbf{n}^T\mathbf{s} \neq \mathbf{0}$, sonst ist die Translation $\mathbf{s}$ parallel zur Ebene E. Im Fall $\mathbf{n}^T\mathbf{s} = \mathbf{0}$ verläuft die Translation von $\mathbf{p}$ entweder in einer Parallelebene von f und es kommt zu keiner Kollision oder sie verläuft in der von f unterstützten Ebene und $\mathbf{p}$ kollidiert mit einer Kante von f. Dieser Fall wird im Rahmen der *Kante–Kante–Kollision* berücksichtigt. Ein Kollisionszeitpunkt wird daher bestimmt durch

$$t_{col}^{trans} = \begin{cases} t & \text{für} \quad t \in [0,1] \text{ und } (\mathbf{p} + t\mathbf{s}) \in f \\ \infty & \text{sonst} \end{cases}$$

Kante–Kante–Kollision

Es ist folgendes Problem zu lösen:

gegeben:

$l_{ab} = \{\mathbf{a} + \mu(\mathbf{b} - \mathbf{a})$	$\mid \quad 0 \leq \mu \leq 1\}$	bewegte Kante als Liniensegment dargestellt
$l_{cd} = \{\mathbf{c} + \nu(\mathbf{d} - \mathbf{c})$	$\mid \quad 0 \leq \nu \leq 1\}$	feste Kante als Liniensegment dargestellt
$\mathbf{s} \in \mathbb{R}^3$		Translationsvektor für l_{ab}

gesucht:

$\exists\, t \in [0,1],\ \mathbf{p} \in l_{ab}, \mathbf{q} \in l_{cd} :\ \mathbf{p} + t\mathbf{s} = \mathbf{q}?$

Das verschobene Liniensegment $l_{ab}(t)$ ergibt sich aus

$$l_{ab}(t) = \{\mathbf{a} + \mu(\mathbf{b} - \mathbf{a}) + t\mathbf{s} \quad | \quad 0 \leq \mu \leq 1\}$$

Damit läßt sich nun der minimale Abstand von l_{ab} und l_{cd} in Richtung $\mathbf{s}$ bestimmen. Dazu löst man das Gleichungssystem

$$\mathbf{a} + \mu(\mathbf{b} - \mathbf{a}) + t\mathbf{s} = \mathbf{c} + \nu(\mathbf{d} - \mathbf{c})$$

mit der Kramerschen Regel. Voraussetzung dafür ist, daß der Translationsvektor $\mathbf{s}$ nicht in der von den beiden Richtungsvektoren $\mathbf{b} - \mathbf{a}$ und $\mathbf{d} - \mathbf{c}$ aufgespannten Ebene liegt und die beiden Liniensegmente l_{ab} und l_{cd} nicht parallel sind. Dies läßt sich zusammenfassen zu dem Test $\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}] \neq 0$. Dann folgt:

$$\begin{aligned} t &= \frac{\det[\mathbf{c} - \mathbf{a}, \mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{d}]}{\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{d}]} \\ \mu &= \frac{\det[\mathbf{s}, \mathbf{c} - \mathbf{a}, \mathbf{c} - \mathbf{d}]}{\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{d}]} \\ \nu &= \frac{\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{a}]}{\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{d}]} \end{aligned}$$

Ein Kollisionszeitpunkt ist dann bestimmt durch

$$t_{col}^{trans} = \begin{cases} t & \text{für} \quad t \in [0,1], \mu \in [0,1], \nu \in [0,1] \\ \infty & \text{sonst} \end{cases}$$

Im Fall, daß $l_{ab} \parallel l_{cd}$ ist ($\equiv \det[\mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}] = 0$), ergibt sich die Möglichkeit eines Schnittes dann, wenn beide Kanten in der von einer Kante und dem Translationsvektor aufgespannten Ebene liegen. Dies kann durch den Test $(\mathbf{s} \times (\mathbf{b} - \mathbf{a}))^T(\mathbf{c} - \mathbf{a}) = 0$ ermittelt werden. In diesem Fall wie in dem Fall, daß der Translationsvektor in der von den beiden Kanten l_{ab} und l_{cd} aufgespannten Ebene liegt, ergibt sich eine Kollision dann, wenn ein Endpunkt einer Kante mit der anderen Kante kollidiert. Sei $\mathbf{p}$ ein Punkt und $l_{uv} = \{\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u}) \mid 0 \leq \alpha \leq 1\}$ ein Liniensegment. Dann kann der Schnittpunkt von $\mathbf{p}$ mit der von dem Liniensegment l_{uv} unterstützten Gerade L_{uv} durch die Lösung des Gleichungssystems

$$\mathbf{p} + t\mathbf{s} = \mathbf{u} + \alpha(\mathbf{v} - \mathbf{u})$$

berechnet werden. Für $0 \leq \alpha \leq 1$ liegt er innerhalb des Liniensegmentes und für $0 \leq t \leq 1$ ergibt sich die Kollision während der Translation $\mathbf{s}$.

Im degenerierten Fall ergibt sich ein Kollisionszeitpunkt zu

$$t_{col}^{trans} = \begin{cases} \min\{t(\{\mathbf{a}, \mathbf{b}\}, l_{cd}), t(\{\mathbf{c}, \mathbf{d}\}, l_{ab})\} & \text{für} \quad ((\det[\mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}] = 0) \wedge \\ & \quad ((\mathbf{s} \times (\mathbf{b} - \mathbf{a}))^T(\mathbf{c} - \mathbf{a}) = 0)) \vee \\ & \quad ((\det[\mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}] \neq 0) \wedge \\ & \quad (\det[\mathbf{s}, \mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}] = 0)) \\ \infty & \quad \text{sonst} \end{cases}$$

2.3.1.2 Kollisionserkennung zwischen zwei Polyedern

Mit Hilfe der elementaren Kollisionsberechnungen Ecke–Fläche und Kante–Kante läßt sich nun der Kollisionstest für zwei Polyeder P_1 und P_2, von denen P_1 um den Vektor $\mathbf{s}$ verschoben wird, kanonisch beschreiben:

$$P_1(t\mathbf{s}) \cap P_2 \neq \emptyset \iff \begin{cases} \exists f_1 \in F_1,\ p_2 \in V_2 & \text{mit} \quad f_1(t\mathbf{s}) \cap p_2 \neq \emptyset \quad \text{oder} \\ \exists p_1 \in V_1,\ f_2 \in F_2 & \text{mit} \quad p_1(t\mathbf{s}) \cap f_2 \neq \emptyset \quad \text{oder} \\ \exists e_1 \in E_1,\ e_2 \in E_2 & \text{mit} \quad e_1(t\mathbf{s}) \cap e_2 \neq \emptyset \end{cases}$$

Dadurch ist der Kollisionstest für zwei Polyeder auf die beschriebenen elementaren Kollisionsberechnungen zurückgeführt. Es ergibt sich folgender Algorithmus:

$$\left.\begin{array}{ll} F_i & \text{Menge aller Flächen von } P_i \\ E_i & \text{Menge aller Kanten von } P_i \\ V_i & \text{Menge aller Eckpunkte von } P_i \\ f_i & \text{Fläche von } P_i \\ e_i & \text{Kante von } P_i \\ \mathbf{v}_i & \text{Eckpunkt von } P_i \end{array}\right\} \quad i = 1, 2$$

$\mathbf{s}$ Translationsvektor
inter Boolesche Variable zur Anzeige einer Kollision

```
inter ← false
forall v1 ∈ V1 do
    forall f2 ∈ F2 do
        if (Ecke-Fläche-Kollision(v1, f2, s)) then
            inter ← true
            goto Ende
forall v2 ∈ V2 do
    forall f1 ∈ F1 do
        if (Ecke-Fläche-Kollision(v2, f1, −s)) then
            inter ← true
            goto Ende
forall e1 ∈ E1 do
    forall e2 ∈ E2 do
        if (Kante-Kante-Kollision(e1, e2, s)) then
            inter ← true
            goto Ende
Ende:
```

Dabei werden alle Ecke-Fläche- und Kante-Kante-Paare der beiden Polyeder gegeneinander getestet und es ergibt sich eine Laufzeit von $O(|P_1| \cdot |P_2|)$.

Für die Kollisionsrechnung bei translatorischen Bewegungen ist jedoch eine Beschleunigung mit Hilfe der hierarchischen Repräsentation konvexer Polyeder möglich.

Effiziente Kollisionsberechnung auf der Basis hierarchischer Repräsentation konvexer Polyeder

Die Verbesserung der Laufzeit bei der Kollisionserkennung ist mit Hilfe der in [DK85] angegebenen hierarchischen Beschreibung eines konvexen Polyeders P möglich. Für die dazu benötigte Datenstruktur wird in [DK85] gezeigt:

Für jedes Polyeder P mit $|P| = n$ gibt es eine Folge $P^{(1)} \subset P^{(2)} \subset \ldots \subset P^{(k)}$ konvexer Polyeder mit folgenden Eigenschaften:

1. $P^{(k)} = P$ und $|P^{(1)}| \leq c_1$,
2. $V^{(i)} \subset V^{(i+1)}$,
3. $V^{(i+1)} \setminus V^{(i)}$ bildet eine unabhängige Knotenmenge in dem Graphen $G^{(i+1)} = (V^{(i+1)}, E^{(i+1)})$, d.h. keine zwei Knoten aus $V^{(i+1)} \setminus V^{(i)}$ sind über eine Kante aus $E^{(i+1)}$ miteinander verbunden.
4. $\max\limits_{i} \max\limits_{v \in V^{(i+1)} \setminus V^{(i)}} \deg_{G^{(i+1)}}(v) \leq c_2$

Die Konstante c_1 steht für die Komplexität des innersten Polyeders der Hierarchie. Während jedes Verfeinerungsschrittes ist die Komplexität der neu hinzukommenden Ecken, Kanten und Flächen durch eine Konstante c_2 beschränkt. Gleichzeitig wird eine Hierarchietiefe von $k = \mathrm{O}(\log n)$ erreicht.

Die Existenz einer solchen logarithmischen Hierarchie beruht auf der Tatsache, daß ein planarer Graph $G = (V, E)$ immer einen konstanten Bruchteil von Knoten mit konstantem Grad $\leq c_2$ besitzt. Unter diesen gibt es wieder einen konstanten Anteil von Knoten, die nicht über Kanten von E direkt benachbart sind. Man kann eine geeignete Knotenmenge mit Hilfe eines Linearzeitalgorithmus finden und damit benötigt der Aufbau der gesamten Hierarchie nur Zeit $O(n)$.

Die hierarchische Repräsentation für das feste Polyeder wird ausgenutzt, um die Abstandsberechnung in Richtung $\mathbf{s}$ zu beschleunigen.

In [Sch94] wird dazu folgendes Lemma bewiesen:

Lemma 2.1 :

Die in eine Translationsrichtung $\mathbf{s}$ *größtmögliche Verschiebung* t_{col}^{trans} *eines Punktes* $\mathbf{q}$, *einer Geraden* G *oder einer Ebene* E *zu einem konvexen Polyeder* P *läßt sich in Zeit* $O(\log |P|)$ *berechnen, wenn* P *hierarchisch repräsentiert ist.*

Es ergibt sich daraus ein Algorithmus zur Bestimmung von $t_{col}^{trans}(P_1, P_2)$:

$t_{col}^{trans} \leftarrow \infty$
forall $f_1 \in F_1$ **do**
  Bestimme einen Punkt $\mathbf{p} \in E(f_1)$ so, daß
  $t_{col}^{trans}(E(f_1), P_2) = t_{col}^{trans}(\mathbf{p}, P_2)$
  if $\mathbf{p} \in f_1$ **then**
    $t_{col}^{trans} \leftarrow \min\{t_{col}^{trans}, t_{col}^{trans}(\mathbf{p}, P_2)\}$
forall $e_1 \in E_1$ **do**
  Bestimme einen Punkt $\mathbf{p} \in G(e_1)$ so, daß
  $t_{col}^{trans}(G(e_1), P_2) = t_{col}^{trans}(\mathbf{p}, P_2)$
  if $\mathbf{p} \in e_1$ **then**
    $t_{col}^{trans} \leftarrow \min\{t_{col}^{trans}, t_{col}^{trans}(\mathbf{p}, P_2)\}$
forall $\mathbf{v}_1 \in V_1$ **do**
  Bestimme $t_{col}^{trans}(\mathbf{v}_1, P_2)$
  $t_{col}^{trans} \leftarrow \min\{t_{col}^{trans}, t_{col}^{trans}(\mathbf{v}_1, P_2)\}$

Dadurch ist folgender Satz bewiesen:

Satz 2.2:

> *Für zwei Polyeder P_1 und P_2, von denen P_2 konvex ist, läßt sich die maximal mögliche Verschiebung $t_{col}^{trans}(P_1, P_2)$ in beliebiger Richtung $\mathbf{s}$ für P_1 in Zeit $O(|P_1| \cdot \log |P_2|)$ berechnen, wenn das konvexe Polyeder P_2 hierarchisch repräsentiert ist.*

Daraus läßt sich in bezug auf die Kollisionsfreiheit einer Translation $\mathbf{s}$ folgendes Korollar ableiten, wenn man beachtet, daß eine Translation $\mathbf{s}$ für ein Polyeder P_1 bzgl. eines festen Polyeders P_2 genau dann kollisionsfrei ist, wenn $t_{col}^{trans}(P_1, P_2) > |\mathbf{s}|$ ist.

Korollar 2.3:

> *Für zwei Polyeder P_1 und P_2, von denen P_2 konvex ist, läßt sich in Zeit $O(|P_1| \cdot \log |P_2|)$ entscheiden, ob P_1 kollisionsfrei bzgl. P_2 um $\mathbf{s}$ verschoben werden kann, wenn P_2 hierarchisch repräsentiert ist.*

Faßt man die Beschreibungskomplexitäten der beiden Polyeder zusammen zu $n = |P_1| + |P_2|$, so ergibt sich eine Gesamtlaufzeit von $O(n \cdot \log n)$. Für den speziellen Fall der Entscheidung, ob eine Kollision auftritt, kann ein Algorithmus angegeben werden, der die gleiche theoretische Laufzeit hat wie der in [Sch94] angegebene.

Eine Kollision zwischen einem translatorisch bewegten Polyeder P_1 und einem Hindernispolyeder P_2 tritt genau dann auf, wenn P_1 und P_2 sich zu einem beliebigen Zeitpunkt der Translation schneiden. D.h. für die Kollisionsentscheidung wird der zeitliche Aspekt der Bewegung vernachlässigt.

Sei nun das bewegte Polyeder P_1 konvex. Betrachtet man die Bewegung von P_1 in einem 4-dimensionalen Objekt-Zeit-Raum, ergibt sich für das Hindernis ein Polyeder, dessen Projektion auf den Objektraum dem Hindernis selbst entspricht. Für das bewegte ergibt sich

Objekt ein Polyeder, dessen Projektion auf den Objektraum dem überstrichenen Volumen Vol_{P_1} entspricht. Eine Vernachlässigung des Zeitaspektes bei einer Bewegung entspricht der Projektion des 4-dimensionalen Objekt-Zeit-Raumes auf den Objektraum, so daß für die Kollisionserkennung die beiden Polyeder Vol_{P_1} und P_2 benutzt werden können. In *Kapitel 4, Abschnitt 4.1.1* wird ein Algorithmus beschrieben, der das überstrichene Volumen eines Polyeders bei translatorischen Bewegungen berechnet. Das überstrichene Volumen eines konvexen Polyeders P_1 ist ein konvexes Polyeder der Größe $O(|P_1|)$ und kann in Zeit $O(|P_1|)$ berechnet werden. In der gleichen Zeit kann die hierarchische Repräsentation von Vol_{P_1} berechnet werden. Mit Hilfe dieser hierarchischen Repräsentation kann in Zeit $O(|P_1| + |P_2| \cdot \log |P_1|)$ entschieden werden, ob Vol_{P_1} und P_2 sich schneiden und damit auch, ob das Polyeder P_1 mit dem Hindernispolyeder P_2 kollidiert. Faßt man die Komplexitäten der beiden Objekte zu $n = |P_1| + |P_2|$ zusammen, ergibt sich eine Laufzeit von $O(n \cdot \log n)$.

Eine weitere Laufzeitverbesserung läßt sich erreichen, wenn man fordert, daß beide Polyeder konvex und hierarchisch repräsentiert sind. Unter dieser Voraussetzung wird in [Sch94] folgender Satz bewiesen:

Satz 2.4:

> *Für zwei konvexe Polyeder P_1 und P_2 läßt sich die größtmögliche Verschiebung $t_{col}^{trans}(P_1, P_2)$ von P_1 in eine Richtung* $\mathbf{s}$ *in Zeit $O(\log |P_1| \cdot \log |P_2|)$ berechnen, wenn P_1 und P_2 hierarchisch repräsentiert sind.*

Berücksichtigt man zusätzlich zu der Laufzeit die Zeit zur Berechnung der hierarchischen Repräsentation von P_1 und P_2, ergibt sich eine Gesamtlaufzeit von $O(|P_1| + |P_2| + \log |P_1| \cdot \log |P_2|)$. Berechnet man bei der oben beschriebenen Vorgehensweise zusätzlich zur hierarchischen Repräsentation von Vol_{P_1} die von P_2, so kann in Zeit $O(\log |P_1| \cdot \log |P_2|)$ entschieden werden, ob eine Kollision zwischen P_1 und P_2 auftritt. Insgesamt ergibt sich also für die Vorgehensweise analog eine Gesamtlaufzeit von $O(|P_1| + |P_2| + \log |P_1| \cdot \log |P_2|)$. Faßt man die beiden Komplexitäten von P_1 und P_2 wieder zusammen zu $n = |P_1| + |P_2|$, so ergibt sich eine Laufzeit von $O(n + \log^2 n)$.

Analog zu Satz 2.2 läßt sich aus der in [Sch94] beschriebenen Vorgehensweise unmittelbar folgendes Korollar ableiten:

Korollar 2.5:

> *Für zwei konvexe Polyeder P_1 und P_2 läßt sich in Zeit $O(\log |P_1| \cdot \log |P_2|)$ entscheiden, ob P_1 kollisionsfrei bzgl. P_2 um einen Vektor* $\mathbf{s}$ *verschoben werden kann, wenn P_1 und P_2 hierarchisch repräsentiert sind.*

Insgesamt lassen sich die Ergebnisse für Translationen in folgendem Satz zusammenfassen:

Satz 2.6 :

Gegeben seien zwei Polyeder P_1 und P_2 sowie ein Translationsvektor $\mathbf{s}$. *Dann läßt sich entscheiden, ob P_1 kollisionsfrei bzgl. P_2 um* $\mathbf{s}$ *verschoben werden kann, in Zeit*

1. $O(|P_1| \cdot |P_2|)$, *falls P_1 und P_2 nicht konvex sind,*
2. $O(k_1 \cdot k_2 \cdot \log |P_1| \cdot \log |P_2|)$, *falls P_1 und P_2 nicht konvex sind, für P_1 eine Zerlegung in k_1 und für P_2 eine Zerlegung in k_2 konvexe Teile bekannt ist und die Teile von P_1 und P_2 hierarchisch repräsentiert sind,*
3. $O(|P_1| \cdot \log |P_2|)$, *falls P_1 nicht konvex und P_2 konvex und hierarchisch repräsentiert ist,*
4. $O(k \cdot \log |P_1| \cdot \log |P_2|)$, *falls P_1 nicht konvex ist, eine Zerlegung in k konvexe Teile bekannt ist und die Teile der Zerlegung hierarchisch repräsentiert sind, sowie P_2 konvex und hierarchisch repräsentiert ist,*
5. $O(\log |P_1| \cdot \log |P_2|)$, *falls P_1 und P_2 konvex und hierarchisch repräsentiert sind.*

2.3.1.3 Translationen bei Polygonen

Sind die Objekte in einer Szene nicht Polyeder im $\mathbb{R}^3$ sondern Polygone im $\mathbb{R}^2$, vereinfacht sich die Kollisionserkennung bei translatorischen Bewegungen. Daß die Kollisionserkennung nicht komplexer ist als im 3-dimensionalen Fall, kann man sich dadurch verdeutlichen, daß das Ergebnis äquivalent ist zu dem Kollisionserkennungsproblem in drei Dimensionen, bei dem die Polygone als Prismenkörper in z-Richtung zu Polyedern erweitert werden. Dabei sind Translationen nur in der x-y-Ebene möglich. Damit ist der 2-dimensionale Fall unter das 3-dimensionale Problem subsumiert.

Eine Betrachtung analog zu Polyedern ergibt, daß zwei Polygone genau dann kollidieren, wenn eine Ecke des einen Polygons mit einer Kante des anderen Polygons kollidiert.

Zur effizienten Kollisionserkennung berechnet man für jede Ecke der beiden Polygone die Kante des anderen Polygons, mit der die Ecke während der Translation als erste kollidiert. Dazu rotiert man die Szene in der Ebene dergestalt, daß die Translation in y-Richtung erfolgt. Danach berechnet man mit Hilfe eines Plane-Sweep-Algorithmus für jede Ecke des bewegten Polygons die Kante des Hindernisses, die in y-Richtung direkt oberhalb der Ecke liegt, sowie für alle Ecken des Hindernispolygons die Kante des bewegten Polygons, die in y-Richtung direkt unterhalb der Ecke liegt. Für den Plane-Sweep-Algorithmus ergeben sich somit folgende Strukturen und Transitionen:

x-Struktur

$$\left.\begin{array}{l}\text{Startpunkte der Kanten}\\ \text{Eckpunkte der Polygone}\end{array}\right\}\ \text{sortiert nach x-Koordinaten}$$

y-Struktur

Kanten der Polygone, die von der aktuellen Sweep-Line geschnitten werden, sortiert nach y-Koordinate des Schnittpunktes

Transitionen

Startpunkt Kante	Einfügen der Kante in die y-Struktur Einfügen des Endpunktes der Kante in die x-Struktur	
Endpunkt Kante	Löschen der Kante aus der y-Struktur	
Eckpunkt Polygon	bewegtes Polygon	Ordne die Ecke der Kante oberhalb der Ecke in der y-Struktur zu, falls sie von einem festen Polygon stammt oder keiner Kante, falls sie vom bewegten Polygon stammt
	festes Polygon	Ordne die Ecke der Kante unterhalb der Ecke in der y-Struktur zu, falls sie von dem bewegten Polygon stammt oder keiner Kante, falls sie von einem festen Polygon stammt

Die Vorgehensweise des Plane-Sweep-Algorithmus bei der Zuordnung von Punkten zu Kanten wird in der Abbildung 2.3 graphisch veranschaulicht.

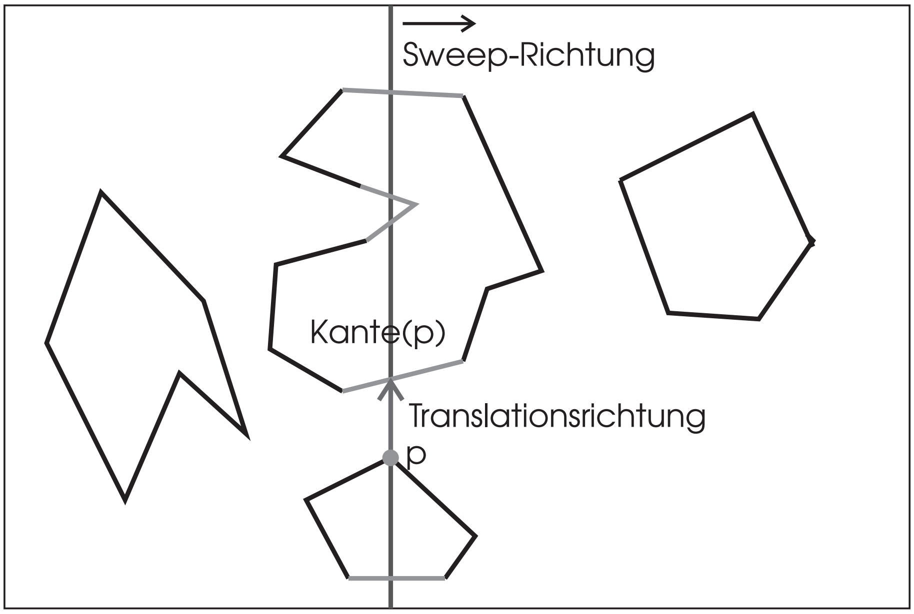

Abbildung 2.3: Plane-Sweep-Algorithmus zur Berechnung von Kollisionen bei Translationen von Polygonen

Jedes durch den Plane-Sweep erzeugte (Ecke,Kante)-Paar definiert einen Abstand in y-Richtung. Ist der dadurch definierte minimale Abstand in y-Richtung kleiner als die Länge der Translation, tritt während der Translation eine Kollision auf.

Das beweist folgenden Satz:

Satz 2.7:

Für zwei Polygone P_1 *und* P_2 *kann in Zeit* $O(n \cdot \log n)$ *entschieden werden, ob* P_1 *während einer Translation mit* P_2 *kollidiert* $(n = |P_1| + |P_2|)$.

Bemerkung 2.8:

Der Algorithmus ist in analoger Weise anwendbar, wenn mehr als ein Polygon in gleicher Weise verschoben wird.

2.3.2 Rotationen

Bei Rotationen ist die gleiche Vorgehensweise möglich wie bei der klassischen Berechnung von Kollisionszeitpunkten bei translatorischen Bewegungen. Eine Verbesserung der Laufzeit analog zu der bei translatorischen Bewegungen gefundenen scheitert. In [ST94] wird dennoch gezeigt, daß subquadratische Algorithmen auch für den Fall von Rotationen existieren.

Bei den betrachteten Rotationen wird angenommen, daß die Rotationsachse durch den Nullpunkt des Koordinatensystems verläuft. Ist dies nicht der Fall, kann dies durch Verschieben der Szene immer erreicht werden.

Formal stellt sich bei einer Rotation folgendes Problem.

gegeben:

P_1	bewegtes Polyeder
P_2	festes Polyeder
$(\mathbf{r}, \theta)$	Rotation von P_1 um normierte Rotationsachse $\mathbf{r}$ um Winkel θ

gesucht:

$\exists\, t \in [\,0, \theta] : \; P_1(\mathbf{r}, t) \cap P_2 \neq \emptyset?$

2.3.2.1 Elementare Kollisionserkennung

Analog zur elementaren Kollisionserkennung bei Translationen läßt sich auch bei der Rotation die Entscheidung, ob die Rotation eines Polyeders P_1 kollisionsfrei bzgl. eines Polyeders P_2 ist, zurückführen auf die Kollisionsarten Ecke–Fläche und Kante–Kante. Die Vorgehensweise wird in [Boy79] anschaulich dargestellt, wobei die darin beschriebenen Formeln zur Problemlösung jedoch nicht kompakt sind, so daß hier eine kompaktere Beschreibung wie in [Sch94] benutzt wird.

Ecke–Fläche–Kollision

Das zu lösende Problem ist nun:
gegeben:

$\mathbf{p} \in \mathbb{R}^3$	Ecke eines Polyeders
f	konvexe Fläche eines Polyeders dargestellt als Polygon im $\mathbb{R}^3$
$(\mathbf{r}, \theta)$	Rotation von $\mathbf{p}$ um normierte Rotationsachse $\mathbf{r}$ um Winkel θ

gesucht:

$\exists\, t \in [\,0, \theta] : \ \mathbf{p}(\mathbf{r}, t) \cap f \neq \emptyset?$

Analog zur Translation wird die Ebenengleichung der von dem Polygon f unterstützten Ebene berechnet als $\mathbf{n}^T\mathbf{x} = d$. In diese wird der rotierende Punkt $\mathbf{p}$ eingesetzt und man erhält folgende Gleichung:

$$\mathbf{n}^T\mathbf{r}\mathbf{r}^T\mathbf{p} + \cos t\,(\mathbf{n}^T\mathbf{p} - \mathbf{n}^T\mathbf{r}\mathbf{r}^T\mathbf{p}) + \sin t\,\mathbf{r}^T(\mathbf{p} \times \mathbf{n}) = d$$

Die allgemeine Form der Gleichung ist

$$\alpha \cos t + \beta \sin t + \gamma = 0$$

wobei

$$\begin{aligned} \alpha &= (\mathbf{n} \times \mathbf{r})^T(\mathbf{p} \times \mathbf{r}) \\ \beta &= \mathbf{p}^T(\mathbf{n} \times \mathbf{r}) \\ \gamma &= \mathbf{n}^T\mathbf{r}\mathbf{r}^T\mathbf{p} - d \end{aligned}$$

Unter Zuhilfenahme der Gleichung

$$\sin^2 t + \cos^2 t = 1$$

ergibt sich als Lösung

$$\begin{aligned} \sin t_{1,2} &= \frac{-\beta\gamma \pm \alpha\sqrt{\alpha^2 + \beta^2 - \gamma^2}}{\alpha^2 + \beta^2} \\ \cos t_{1,2} &= \frac{-\alpha\gamma \mp \beta\sqrt{\alpha^2 + \beta^2 - \gamma^2}}{\alpha^2 + \beta^2} \end{aligned}$$

Aus dieser Lösung lassen sich die beiden Kollisionswinkel t_1 und $t_2 \in [\,0, 2\pi]$ berechnen. Eine Kollision ergibt sich dann, wenn $t_i \leq \theta$ und $\mathbf{p}(\mathbf{r}, t_i) \in f$. Dies läßt sich ausdrücken als

$$t_{col}^{rot} = \begin{cases} \min_{i=1,2}\{t_i \mid 0 \leq t_i \leq \theta,\ \mathbf{p}(\mathbf{r}, t_i) \in f\} & \text{falls } t_1 \text{ oder } t_2 \text{ existiert} \\ \infty & \text{sonst} \end{cases}$$

Falls die Rotationsachse orthogonal zu der von f unterstützten Ebene liegt, d.h. $\mathbf{n} \times \mathbf{r} = 0$, bewegt sich $\mathbf{p}$ in einer Ebene parallel zu der von f unterstützten ($\gamma \neq 0$) – dadurch kann keine Kollision auftreten – oder $\mathbf{p}$ bewegt sich in der von f unterstützten Ebene ($\gamma = 0$) und $\mathbf{p}$ kann mit einer Kante von f kollidieren. Dieser Fall wird bei der Betrachtung der Kante–Kante–Kollision erfaßt.

Kante–Kante–Kollision

gegeben:

$l_{ab} = \{\mathbf{a} + \mu(\mathbf{b} - \mathbf{a}) \mid 0 \leq \mu \leq 1\}$	bewegte Kante beschrieben als Liniensegment
$l_{cd} = \{\mathbf{c} + \nu(\mathbf{d} - \mathbf{c}) \mid 0 \leq \nu \leq 1\}$	feste Kante beschrieben als Liniensegment
$(\mathbf{r}, \theta)$	Rotation für l_{ab} um normierte Rotationsachse $\mathbf{r}$ um Winkel θ

gesucht:

$\exists\, t \in [\,0, \theta] : \; l_{ab}(\mathbf{r}, \theta) \cap l_{cd} \neq \emptyset?$

Das rotierte Liniensegment $l_{ab}(t)$ läßt sich darstellen als

$$\mathbf{R}(t)\,(\mathbf{a} + \mu(\mathbf{b} - \mathbf{a})) \quad \text{mit} \quad 0 \leq t \leq \theta,\; 0 \leq \mu \leq 1$$

wobei $\mathbf{R}(t)$ die zu der Rotation um die Achse $\mathbf{r}$ und den Winkel t gehörige Rotationsmatrix ist. Dadurch läßt sich die Berechnung eines Kollisionszeitpunktes zwischen $l_{ab}(\mathbf{r}, t)$ und l_{cd} zurückführen auf die Lösung der Gleichung

$$\mathbf{R}(t)\,(\mathbf{a} + \mu(\mathbf{b} - \mathbf{a})) = \mathbf{c} + \nu(\mathbf{d} - \mathbf{c}) \quad \text{mit} \quad 0 \leq t \leq \theta,\; 0 \leq \mu, \nu \leq 1 \tag{2.1}$$

Diese Gleichung ist ein nichtlineares Gleichungssystem mit den drei Unbekannten t, μ und ν. Um dieses lösen zu können, geht man zunächst von den Liniensegmenten l_{ab} und l_{cd} zu den zugehörigen Geraden L_{ab} und L_{cd} über und betrachtet, wann sich diese schneiden.
Für zwei Geraden

$$\begin{aligned} L_{ab} &= \{\mathbf{a} + \mu(\mathbf{b} - \mathbf{a}) \mid \mu \in \mathbb{R}\} \\ L_{cd} &= \{\mathbf{c} + \nu(\mathbf{d} - \mathbf{c}) \mid \nu \in \mathbb{R}\} \end{aligned}$$

gilt:

$$L_{ab} \cap L_{cd} \neq \emptyset \iff L_{ab} \not\parallel L_{cd} \wedge \mathbf{d} \in E_{\mathbf{a},\mathbf{b}-\mathbf{a},\mathbf{d}-\mathbf{c}}$$

wobei

$$E_{\mathbf{a},\mathbf{b}-\mathbf{a},\mathbf{d}-\mathbf{c}} = \{\mathbf{x} \in \mathbb{R}^3 \mid ((\mathbf{b} - \mathbf{a}) \times (\mathbf{d} - \mathbf{c}))^T (\mathbf{x} - \mathbf{a}) = 0\}$$

die von dem Punkt $\mathbf{a}$ und den beiden Richtungsvektoren $\mathbf{b} - \mathbf{a}$ und $\mathbf{d} - \mathbf{c}$ aufgespannte Ebene ist.

Betrachten wir nun die beiden Bedingungen genauer, so ergibt sich

$$\begin{array}{lrcl}
(1) & L_{ab} \not\parallel L_{cd} & \iff & ((\mathbf{b} - \mathbf{a}) \times (\mathbf{d} - \mathbf{c})) \neq \mathbf{0} \\
(2) & d \in E_{\mathbf{a},\mathbf{b}-\mathbf{a},\mathbf{d}-\mathbf{c}} & \iff & ((\mathbf{b} - \mathbf{a}) \times (\mathbf{d} - \mathbf{c}))^T (\mathbf{d} - \mathbf{a}) = 0 \\
 & & \iff & \det[\mathbf{b} - \mathbf{a}, \mathbf{d} - \mathbf{c}, \mathbf{d} - \mathbf{a}] = 0 \\
 & & \iff & \det[\mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{a}, \mathbf{d} - \mathbf{a}] = 0 \\
 & & \iff & \det[\mathbf{b} - \mathbf{a}, \mathbf{c}, \mathbf{d}] + \det[\mathbf{a}, \mathbf{b}, \mathbf{d} - \mathbf{c}] = 0 \\
 & & \iff & (\mathbf{c} \times \mathbf{d})^T (\mathbf{b} - \mathbf{a}) + (\mathbf{d} - \mathbf{c})^T (\mathbf{a} \times \mathbf{b}) = 0
\end{array}$$

Setzt man nun die Werte von $L_{ab}(\mathbf{r}, t)$ und L_{cd} in die beiden Bedingungen ein, erhält man

$$\begin{array}{lrcl}
(1) & ((\mathbf{R}(t)(\mathbf{b} - \mathbf{a}) \times (\mathbf{d} - \mathbf{c})) & \neq & \mathbf{0} \\
(2) & (\mathbf{c} \times \mathbf{d})^T \mathbf{R}(t)(\mathbf{b} - \mathbf{a}) + (\mathbf{d} - \mathbf{c})^T \mathbf{R}(t)(\mathbf{a} \times \mathbf{b}) & = & 0
\end{array}$$

Mit der Definition der Rotationsmatrix $\mathbf{R}(t)$ läßt sich folgende Identität verifizieren:

$$\mathbf{q}^T\mathbf{R}(t)\mathbf{p} = \mathbf{q}^T\mathbf{r}\mathbf{r}^T\mathbf{p} + \cos t\ (\mathbf{q}^T\mathbf{p} - \mathbf{q}^T\mathbf{r}\mathbf{r}^T\mathbf{p}) + \sin t\ \mathbf{r}^T(\mathbf{p}\times\mathbf{q})$$

Daraus ergibt sich, daß die zweite Bedingung eine Gleichung der Form

$$\alpha\ \cos t + \beta\ \sin t + \gamma = 0$$

ist mit

$$\begin{aligned} \alpha &= (\mathbf{d}-\mathbf{c})^T(\mathbf{a}\times\mathbf{b}) - \mathbf{r}^T(\mathbf{d}-\mathbf{c})\mathbf{r}^T(\mathbf{a}\times\mathbf{b}) \\ &\quad +(\mathbf{b}-\mathbf{a})^T(\mathbf{c}\times\mathbf{d}) - \mathbf{r}^T(\mathbf{b}-\mathbf{a})\mathbf{r}^T(\mathbf{c}\times\mathbf{d}) \\ \beta &= \mathbf{r}^T((\mathbf{b}-\mathbf{a})\times(\mathbf{c}\times\mathbf{d}) - (\mathbf{d}-\mathbf{c})\times(\mathbf{a}\times\mathbf{b})) \\ \gamma &= \mathbf{r}^T(\mathbf{d}-\mathbf{c})\mathbf{r}^T(\mathbf{a}\times\mathbf{b}) + \mathbf{r}^T(\mathbf{b}-\mathbf{a})\mathbf{r}^T(\mathbf{c}\times\mathbf{d}) \end{aligned}$$

Diese besitzt wie oben bereits beschrieben die beiden Lösungen

$$\begin{aligned} \sin t_{1,2} &= \frac{-\beta\gamma \pm \alpha\sqrt{\alpha^2+\beta^2-\gamma^2}}{\alpha^2+\beta^2} \\ \cos t_{1,2} &= \frac{-\alpha\gamma \mp \beta\sqrt{\alpha^2+\beta^2-\gamma^2}}{\alpha^2+\beta^2} \end{aligned}$$

unter der Voraussetzung, daß nicht $\alpha = \beta = \gamma = 0$. Dies tritt dann ein, wenn

(1) beide Geraden parallel zur Rotationsachse sind $((\mathbf{r} \parallel \mathbf{b}-\mathbf{a}) \wedge (\mathbf{r} \parallel \mathbf{d}-\mathbf{c}))$,

(2) beide Geraden senkrecht zur Rotationsachse sind $((\mathbf{r} \perp \mathbf{b}-\mathbf{a}) \wedge (\mathbf{r} \perp \mathbf{d}-\mathbf{c}))$ oder

(3) beide Geraden einen gemeinsamen Schnittpunkt auf der Rotationsachse haben.

Im Fall (1) kann eine Kollision auftreten, wenn L_{ab} und L_{cd} denselben Abstand zur Rotationsachse haben. Betrachtet man die Einschränkung der Geraden auf die Kanten, sieht man, daß in diesem Fall eine Kollision einer zu l_{ab} adjazenten Kante mit l_{cd} in einem nichtentarteten Fall auftritt. Der Fall (2) kann analog zu Fall (1) auf eine Kollision in einem nichtentarteten Fall zurückgeführt werden, falls eine Kollision vorliegt. Liegt im Fall (3) der Schnittpunkt der Geraden innerhalb der Kanten, ergibt sich eine Kollision zum Zeitpunkt $t = 0$.

Die beiden Kollisionswinkel können nun in die Gleichung 2.1 eingesetzt werden und man erhält ein lösbares lineares Gleichungssystem.

Durch die Lösung dieses Gleichungssystems läßt sich nun entscheiden, ob während der Rotation eine Kollision auftritt durch

$$t_{col}^{rot} = \begin{cases} \min_{i=1,2}\{t_i \mid 0 \le t_i \le \theta,\ 0 \le \mu \le 1,\ 0 \le \nu \le 1\} & \text{falls } t_1 \text{ oder } t_2 \text{ existiert} \\ \infty & \text{sonst} \end{cases}$$

Analog zu der bei den Translationen beschriebenen Vorgehensweise erhält man für die Rotation folgenden Algorithmus:

F_i	Menge aller Flächen von P_i	$i = 1, 2$
E_i	Menge aller Kanten von P_i	
V_i	Menge aller Eckpunkte von P_i	
f_i	Fläche von P_i	
e_i	Kante von P_i	
$\mathbf{v}_i$	Eckpunkt von P_i	
$\mathbf{r}$	normierte Rotationsachse	
θ	Rotationswinkel	
inter	Boolesche Variable zur Anzeige einer Kollision	

```
inter ← false
forall v1 ∈ V1 do
    forall f2 ∈ F2 do
        if (Ecke-Fläche-Kollision(v1, f2, r, θ)) then
            inter ← true
            goto Ende
forall v2 ∈ V2 do
    forall f1 ∈ F1 do
        if (Ecke-Fläche-Kollision(v2, f1, −r, θ)) then
            inter ← true
            goto Ende
forall e1 ∈ E1 do
    forall e2 ∈ E2 do
        if (Kante-Kante-Kollision(e1, e2, r, θ)) then
            inter ← true
            goto Ende
Ende:
```

Die Ergebnisse für den Fall von Rotationen lassen sich zusammenfassen durch folgenden Satz:

Satz 2.9 :

> *Für zwei beliebige Polyeder P_1 und P_2 und eine Rotation für P_1, gegeben durch eine normierte Rotationsachse* $\mathbf{r}$ *durch den Nullpunkt und einen Rotationswinkel θ, kann in Zeit $O(|P_1| \cdot |P_2|)$ entschieden werden, ob P_1 und P_2 während der Rotation miteinander kollidieren.*

In [ST94] wird gezeigt, daß die Kollisionserkennung bei Rotationen für Polyeder auch in subquadratischer Zeit möglich ist. Ein praktikabler subquadratischer Algorithmus ist jedoch bisher nicht bekannt.

2.3.2.2 Rotationen bei Polygonen

Die Rotation von Polygonen kann analog zu der Vorgehensweise bei Translationen gehandhabt werden. Jeder Ecke eines Polygons wird die Kante des jeweils anderen Polygons zugeordnet, mit der diese Ecke als erste während der Rotation kollidiert. Dazu führt man - wie bei der Translation - einen Sweep orthogonal zur Bewegungsrichtung durch. Sei O.B.d.A. das Rotationszentrum der Nullpunkt des Koordinatensystems, sonst verschiebt man die Szene so, daß das Rotationszentrum im Nullpunkt liegt.

Betrachtet man sich die Darstellung der Polygone im Polarkoordinatensystem, d.h. jeder Punkt wird durch den Abstand zu dem Rotationszentrum und den Winkel φ mit einem Referenzstrahl ausgehend vom Rotationszentrum beschrieben, entspricht die Rotation eines Punktes um den Rotationspunkt einer Translation des Punktes in φ-Richtung, da sich der Abstand zum Rotationszentrum während einer Rotation nicht ändert. Diese Darstellung der Szene bietet sich somit als Basis für einen Plane-Sweep in Abstandsrichtung ausgehend vom Rotationszentrum an. Übertragen in das euklidische Koordinatensystem entspricht dies einem Sweep, bei dem die Sweep-Line ein sich ausdehnender Kreis mit Mittelpunkt im Rotationszentrum ist.

Bei der Übertragung der Kanten in das Polarkoordinatensystem muß man zusätzlich beachten, daß diese zu Hyperbelbögen werden, deren minimaler Abstand zum Rotationszentrum nicht in einem der beiden Endpunkte der Kante liegen muß. Der Minimalpunkt liegt im Lotfußpunkt der von der Kante unterstützten Gerade bzgl. des Rotationspunktes, falls er sich innerhalb der Kante befindet. Dieser läßt sich berechnen durch

$$\mathbf{l} = \mathbf{a} + \frac{\mathbf{a}^T(\mathbf{a}-\mathbf{b})}{|\mathbf{b}-\mathbf{a}|^2}(\mathbf{b}-\mathbf{a}) \text{ für } e = \{\mathbf{x} \mid \mathbf{x} = \mathbf{a} + \mu(\mathbf{b}-\mathbf{a}),\ 0 \leq \mu \leq 1\}$$

und er liegt innerhalb der Kante, falls

$$0 < \frac{\mathbf{a}^T(\mathbf{a}-\mathbf{b})}{|\mathbf{b}-\mathbf{a}|^2} < 1$$

gilt. Als Punkt mit maximalem Abstand kommen nur die beiden Endpunkte der Kante in Frage.

Allerdings schneiden sich die Hyperbelbögen nicht außer in ihren Endpunkten, da ein Schnittpunkt zwischen zwei Hyperbelbögen im Polarkoordinatensystem eindeutig einem Punkt im kartesischen Koordinatensystem entspricht. Dieser läge auf beiden Kanten und ist kein gemeinsamer Endpunkt, was ein Widerspruch zur Voraussetzung ist, daß sich zwei Kanten höchstens in ihren Endpunkten schneiden.

Das Polarkoordinatensystem wird implizit bei der Beschreibung des Sweepes im euklidischen Koordinatensystem benutzt.

Damit die Minimalpunkte Endpunkte der Kanten der Polygone sind, werden Kanten, deren Minimalpunkt im Inneren der Kante liegt, an dieser Stelle zerlegt, so daß der Minimalpunkt wieder ein Endpunkt der Kante ist. Dadurch wird jede Kante höchstens in zwei Teile zerlegt, so daß sich die Komplexität des betrachteten Problems nicht ändert. Durch den Abstand der beiden Endpunkte einer Kante zum Rotationszentrum wird für jede Kante ein Start- und ein Endpunkt definiert.

Im folgenden ist der Plane–Sweep–Algorithmus im einzelnen erläutert:

x-Struktur

$$\left.\begin{array}{l}\text{Startpunkte der Kanten}\\ \text{Eckpunkte der Polygone}\end{array}\right\}\text{ sortiert nach Abstand zum Rotationszentrums}$$

y-Struktur

Kanten, die von der aktuellen Sweep-Line geschnitten werden, zyklisch sortiert nach ihrem Winkel mit einem Referenzstrahl ausgehend vom Rotationszentrum

Transitionen

Startpunkt Kante	Einfügen der Kante in die y-Struktur Einfügen des Endpunktes der Kante in die x-Struktur	
Endpunkt Kante	Löschen der Kante aus der y-Struktur	
Eckpunkt Polygon	bewegtes Polygon	Ordne die Ecke der Kante zu, die in der y-Struktur zyklisch oberhalb der Ecke in der y-Struktur liegt, falls diese von einem festen Polygon stammt, oder keiner Kante
	festes Polygon	Ordne die Ecke der Kante zu, die in der y-Struktur zyklisch unterhalb der Ecke in der y-Struktur liegt, falls diese vom bewegten Polygon stammt, oder keiner Kante

Die Vorgehensweise des Algorithmus bei der Zuordnung der Ecken zu Kanten ist in Abbildung 2.4 graphisch dargestellt.

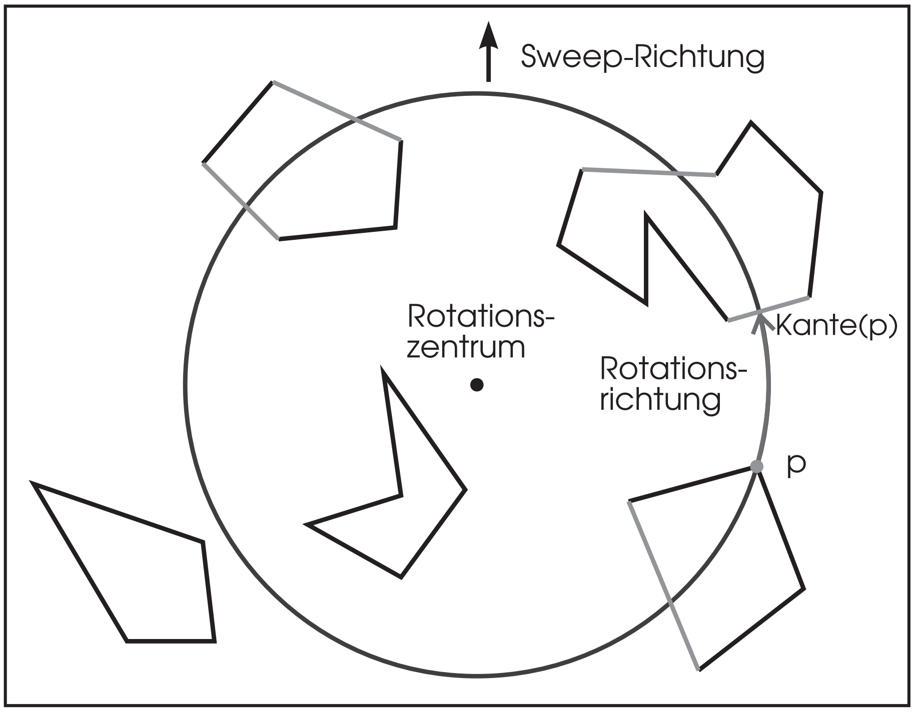

Abbildung 2.4: Circle-Sweep-Algorithmus zur Berechnung von Kollisionen bei Rotationen von Polygonen in der Ebene

Bei den Eckpunkten, deren in jeweiliger Richtung benachbarte Kante vom gleichen Polygon stammt, ist keine Zuordnung durchzuführen, da diese Ecke die erste Kollision nicht verursachen kann.

Sei P_1 das bewegte und P_2 das feste Polygon. Aus den durch den Sweep gefundenen (Ecke,Kante)-Paaren ergibt sich der maximale kollisionsfreie Rotationswinkel zu

$$\min\{min\{\varphi(\mathbf{v}, edge(\mathbf{v}) \mid \mathbf{v} \in P_1\},\ min\{\varphi(edge(\mathbf{v}), \mathbf{v} \mid \mathbf{v} \in P_2\}\}$$

wobei $\varphi(a, b)$ der maximale kollisionsfreie Winkel für a und b bei einer Rotation um das gegebene Rotationszentrum ist. Ist der maximal mögliche Rotationswinkel kleiner als die durchgeführte Rotation, ergibt sich eine Kollision.

Diese Vorgehensweise beweist die folgende Aussage:

Satz 2.10 :

Für zwei Polygone P_1 und P_2 kann in Zeit $O(n \cdot \log n)$ entschieden werden, ob P_1 während einer Rotation mit P_2 kollidiert ($n = |P_1| + |P_2|$).

Bemerkung 2.11 :

Der Algorithmus ist in analoger Vorgehensweise anwendbar, wenn mehr als ein Polygon in gleicher Weise rotiert wird.

2.3.3 Beschleunigung durch Heuristiken

Während in den vorhergehenden Abschnitten Algorithmen beschrieben wurden, mit denen Kollisionsen bei Bewegungen berechnet werden können, bei denen stets alle Hindernisse als relevant für eine Kollision angenommen und untersucht wurden, soll in diesem Abschnitt auf Heuristiken eingegangen werden, die die praktische Laufzeit der Kollisionserkennung reduzieren helfen. Grundsätzlich lassen sich diese Heuristiken in zwei Kategorien einteilen:

1. Reduktion der Anzahl der zu betrachtenden Hindernisse

2. Reduktion der Anzahl der zu betrachtenden Objektteile

Ein Überblick über bekannte Techniken kann [Sch94] entnommen werden; nähere Erläuterungen, praktische Implementierungen und Messungen über das Laufzeitverhalten einzelner Heuristiken sind in [Fr93] zu finden.

An dieser Stelle sollen nun einige dieser Heuristiken beschrieben werden, die zusammen eine sinnvolle Hierarchie zur Berechnung von Kollisionen zwischen Polyedern im $\mathbb{R}^3$ bzw. Polygonen im $\mathbb{R}^2$ ergeben.

2.3.3.1 Elimination von Hindernissen

Ziel der Elimination von Hindernissen ist es, in einer Vorberechnung die Anzahl der Hindernisse, die einer näheren Kollisionsbetrachtung bedürfen, durch einfache Tests möglichst stark zu reduzieren. Insbesondere bei großen Szenen ist diese Art der Vorberechnung besonders effizienzsteigernd.

Kugelheuristik

Die grundlegende Idee für diese Heuristik ist:

> Wenn sich bei einer Bewegung die umhüllenden Kugeln eines bewegten Objektes und eines festen Hindernisses nicht schneiden, kollidiert das Objekt mit dem Hindernis während der Bewegung nicht.

Dazu wird für jedes Hindernis und das zu bewegende Objekt die kleinste einschließende Kugel berechnet. Dies erfolgt im $\mathbb{I}\mathbb{G}\mathbb{O}_{\mathbb{R}}$ -System nicht analytisch, sondern mit Hilfe eines numerischen Verfahrens zur konvexen Programmierung, des *Downhill Simplex Algorithmus* von Nelder und Mead [NM65]. Dazu wird die nicht differenzierbare Funktion

$$\max_i |\mathbf{p}_i - \mathbf{c}| \quad \text{mit} \quad \mathbf{p}_i \in Eckenmenge, \ \mathbf{c} \in \mathbb{R}^3$$

über $\mathbf{c}$ minimiert. $\mathbf{c}$ definiert den Mittelpunkt der Kugel und $\max_i |\mathbf{p_i} - \mathbf{c}|$ den Radius. Mit Hilfe von Mittelpunkt und Radius zweier Kugeln K_1 und K_2 ergibt sich deren euklidischer Abstand

$$\mathrm{d}(K_1, K_2) = |\mathbf{c}_2 - \mathbf{c}_1| - (r_1 + r_2)$$

Zwei Kugeln schneiden sich, wenn ihr Abstand kleiner gleich null ist, d.h.

$$|\mathbf{c}_2 - \mathbf{c}_1| - (r_1 + r_2) \leq 0$$

Für alle Paare (K_{bewegt}, K_{fest}), die sich schneiden, muß eine exaktere Kollisionsrechnung durchgeführt werden. Die anderen Paare können heuristisch weiterbearbeitet werden.

Translationen

Bei allen Paaren (K_{bewegt}, K_{fest}) von Kugeln, deren euklidischer Abstand größer als die Länge der Translation ist, kann keine Kollision auftreten.
Für alle anderen Paare wird getestet, ob während einer Translation $\mathbf{s}$ eine Kollision auftritt. Dies ist der Fall, wenn sich die beiden Kugeln berühren, d.h.

$$\begin{aligned}
& & |(\mathbf{c}_b + t\mathbf{s}) - \mathbf{c}_f| &= r_b + r_f \\
\iff & & (\mathbf{c}_b + t\mathbf{s} - \mathbf{c}_f)^T(\mathbf{c}_b + t\mathbf{s} - \mathbf{c}_f) &= (r_b + r_f)^2 \\
\iff & & t^2 + 2\tfrac{(\mathbf{c}_b - \mathbf{c}_f)^T\mathbf{s}}{|\mathbf{s}|^2}t + \tfrac{(\mathbf{c}_b - \mathbf{c}_f)^T(\mathbf{c}_b - \mathbf{c}_f) - (r_b + r_f)^2}{|\mathbf{s}|^2} &= 0 \\
\iff & & t_{1,2} = -\tfrac{(\mathbf{c}_b - \mathbf{c}_f)^T\mathbf{s}}{|\mathbf{s}|^2} \pm \sqrt{\left(\tfrac{(\mathbf{c}_b - \mathbf{c}_f)^T\mathbf{s}}{|\mathbf{s}|^2}\right)^2 - \tfrac{(\mathbf{c}_b - \mathbf{c}_f)^T(\mathbf{c}_b - \mathbf{c}_f) - (r_b + r_f)^2}{|\mathbf{s}|^2}} & &
\end{aligned}$$

Ist der Radikant des Wurzelausdrucks negativ, so ergibt sich keine Kollision. Eine Kollision ergibt sich nur dann, wenn $0 < t_i < 1$ ist, da dann die Translation $\mathbf{s}$ nicht vollständig durchgeführt werden kann. Für diese Paare (K_{bewegt}, K_{fest}) müssen exaktere Kollisionstests durchgeführt werden.

Rotationen

Eine rotierende Kugel K_{bewegt} kollidiert mit einer festen Kugel K_{fest} während einer Rotation um eine Rotationsachse $\mathbf{r}$ und Rotationswinkel θ, falls

$$\begin{aligned}
& |\mathbf{R}(t)\mathbf{c}_b - \mathbf{c}_f| &=& \; r_b + r_f \\
\Longleftrightarrow \quad & (\mathbf{R}(t)\mathbf{c}_b - \mathbf{c}_f)^T(\mathbf{R}(t)\mathbf{c}_b - \mathbf{c}_f) &=& \; (r_b + r_f)^2 \\
\Longleftrightarrow \quad & -2\mathbf{c}_f^T(\mathbf{c}_b - (\mathbf{r}^T\mathbf{c}_b)\mathbf{r})\cos t & & \\
& -2\mathbf{c}_f^T(\mathbf{r} \times \mathbf{c}_b)\sin t & & \\
& +|\mathbf{c}_b|^2 + |\mathbf{c}_f|^2 - (r_b + r_f)^2 - 2\mathbf{c}_f^T(\mathbf{r}^T\mathbf{c}_b)\mathbf{r} &=& \; 0
\end{aligned}$$

Dies ist eine Gleichung der Form

$$\alpha \cos t + \beta \sin t + \gamma = 0$$

mit

$$\begin{aligned}
\alpha &= -2\mathbf{c}_f^T(\mathbf{c}_b - (\mathbf{r}^T\mathbf{c}_b)\mathbf{r}) \\
\beta &= -2\mathbf{c}_f^T(\mathbf{r} \times \mathbf{c}_b) \\
\gamma &= |\mathbf{c}_b|^2 + |\mathbf{c}_f|^2 - (r_b + r_f)^2 - 2\mathbf{c}_f^T(\mathbf{r}^T\mathbf{c}_b)\mathbf{r}
\end{aligned}$$

so daß sich die Lösung zu

$$\begin{aligned}
\sin t_{1,2} &= \frac{-\beta\gamma \pm \alpha\sqrt{\alpha^2 + \beta^2 - \gamma^2}}{\alpha^2 + \beta^2} \\
\cos t_{1,2} &= \frac{-\alpha\gamma \mp \beta\sqrt{\alpha^2 + \beta^2 - \gamma^2}}{\alpha^2 + \beta^2}
\end{aligned}$$

ergibt.

Ist der Radikand des Wurzelausdrucks negativ, ergibt sich keine Kollision. Eine Kollision ergibt sich nur dann, wenn $0 \leq t_i \leq \theta$ ist, da dann die Rotation nicht vollständig durchgeführt werden kann. Für diese Paare von Kugeln muß analog zur Translation eine exaktere Kollisionsrechnung durchgeführt werden.

2.3.3.2 Elimination von Objektteilen

Bei der Elimination von Objektteilen sollen elementare Kollisionsbetrachtungen zwischen einem Polyeder P_{bewegt} und einem festen Polyeder P_{fest} eingespart und dadurch die gesamte Kollisionserkennung beschleunigt werden.

Elimination zu weit entfernter Flächen

Bei der Elimination zu weit entfernter Flächenpaare sollen Kollisionen zwischen zwei Flächen ausgeschlossen werden, deren umhüllenden Kugeln während einer Bewegung nicht miteinander kollidieren.

Das zu lösende Problem läßt sich wie folgt formalisieren:
gegeben:

f_{bewegt}	bewegte Fläche
f_{fest}	feste Fläche
w	Bewegung für f_{bewegt}

gesucht:

$\exists\, t \in [\,0,1]: \; K(f_{bewegt}(w(t))) \cap K(f_{fest}) \neq \emptyset?$

Dazu werden für die beiden Flächen die kleinsten umhüllenden Kugeln berechnet. Dies kann analog zur Kugelheuristik numerisch mit Hilfe des *Downhill Simplex Algorithmus* von Nelder und Mead [NM65] erfolgen.

Für Translationen und Rotationen kann nun dieselbe Vorgehensweise wie bei der Betrachtung kleinster umhüllender Kugeln für ganze Polyeder erfolgen. Dadurch kann für ein Paar von Flächen entschieden werden, ob sie bei der gegebenen Bewegung kollidieren können. Für diese Paare muß eine exaktere Kollisionserkennung durchgeführt werden.

Durch diese Heuristiken wird eine Hierarchie zur Berechnung von Kollisionen zwischen zwei Polyedern beschrieben, die auf der elementaren Kollisionserkennung für Paare $(Ecke, Fläche)$ und $(Kante, Kante)$ basiert. Zusammengefaßt ergibt sich folgender Algorithmus zur Berechnung der Kollisionen zwischen einem bewegten Polyeder und einer Menge von festen Polyedern:

P_{bewegt}	bewegtes Polyeder
F_{bewegt}	Menge der Flächen des bewegten Polyeders
f_{bewegt}	Fläche des bewegten Polyeders
e_{bewegt}	Kante des bewegten Polyeders
v_{bewegt}	Ecke des bewegten Polyeders
H	Menge der Hindernispolyeder
P_{fest}	Hindernispolyeder
F_{fest}	Menge der Flächen des festen Polyeders P_{fest}
f_{fest}	Fläche des festen Polyeders P_{fest}
e_{fest}	Kante des festen Polyeders P_{fest}
v_{fest}	Ecke des festen Polyeders P_{fest}
Weg	Liste von Translationen und Rotationen als Bewegung für P_{bewegt}
b	Einzelbewegung des $Weges$ mit $b \in \{Translation, Rotation\}$
$Kollision$	Boolesche Variable zur Anzeige einer Kollision

Vorberechnet sind:

$K(P_{bewegt})$
$\forall f_{bewegt} \in F_{bewegt} : K(f_{bewegt})$
$\forall P_{fest} \in H : \; K(P_{fest})$ und $\forall f_{fest} \in F_{P_{fest}} : \; K(f_{fest})$

```
Kollision ← false
forall b ∈ Weg do
    forall P_fest ∈ H do
        /* 1. Stufe: Elimination von Hindernissen mit der Kugelheuristik */
        if (Kollision(K(P_bewegt), K(P_fest), b) = true) then
            forall f_bewegt ∈ F_bewegt do
                forall f_fest ∈ F_fest do
                    /* 2. Stufe: Genauere Kugelheuristik für Flächenpaare */
                    if (Kollision(K(f_bewegt), K(f_fest), b) = true) then
                        /* 3. Stufe: Exakte Kollisionsrechnung für Flächenpaare */
                        forall v_bewegt ∈ f_bewegt do
                            if (Kollision(v_bewegt, f_fest, b) = true) then
                                Kollision ← true
                                goto Ende
                        forall v_fest ∈ f_fest do
                            if (Kollision(v_fest, f_bewegt, −b) = true) then
                                Kollision ← true
                                goto Ende
                        forall e_bewegt ∈ f_bewegt do
                            forall e_fest ∈ f_fest do
                                if (Kollision(e_bewegt, e_fest, b) = true) then
                                    Kollision ← true
                                    goto Ende
Ende:
```

2.3.3.3 Heuristiken in der Ebene

Die Kugelheuristik für Polyeder kann analog als Kreisheuristik für Polygone durchgeführt werden.

2.3.4 Zusammenfassung

In diesem Abschnitt wurden Kollisionserkennungsalgorithmen für translatorische und rotatorische Bewegungen eines bewegten Polyeders bzw. Polygons und einer Menge von festen Hindernispolyedern bzw. -polygonen beschrieben.

Für Polyeder wurden bei Translationen neben dem klassischen $O(n^2)$ Algorithmus für den Fall beliebiger Polyeder weitere effiziente Verfahren zur Kollisionserkennung beschrieben. Sind alle beteiligten Polyeder konvex, wurde ein $O(n)$ Algorithmus beschrieben und, falls die hierarchischen Repräsentationen der Polyeder bereits berechnet sind, auf einen $O(\log^2 n)$ in [Sch94] verwiesen. Ist das bewegte Polyeder oder sind die Hindernispolyeder nicht konvex, wurde ein $O(n \cdot \log n)$ angegeben. Für den Fall nichtkonvexer Polyeder, bei dem für das bewegte Polyeder oder die Hindernispolyeder eine Zerlegung in k konvexe Teile bekannt ist, die alle

hierarchisch repräsentiert sind, ergibt sich daraus ein $O(k \cdot n \cdot \log n)$ Algorithmus.
Für Rotationen wurde der klassische $O(n^2)$ Algorithmus beschrieben; eine Übertragung der effizienten Techniken für Translationen auf Rotationen ist nicht möglich. Jedoch wird in [ST94] gezeigt, daß eine Kollisionserkennung für zwei beliebige Polyeder bei Translationen in Zeit $O(n^{8/5+\epsilon})$ und bei Rotationen in Zeit $O(n^{5/3+\epsilon})$ möglich ist, wobei ϵ eine beliebige positive Konstante ist.

Für beliebige Polygone wurde im Falle von Translationen und Rotationen ein $O(n \cdot \log n)$ Algorithmus entwickelt.

Zusätzlich zu diesen exakten Algorithmen wurde eine Hierarchie von Heuristiken beschrieben, mit deren Hilfe die Kollisionserkennung bei praktischen Problemen beschleunigt werden kann. Weitere Techniken zur Kollisionserkennung können [Sch94] und [Fr93] entnommen werden.

Insgesamt kann nun für ein bewegtes Polyeder bzw. Polygon, eine Menge von Hindernispolyedern bzw. -polygonen und eine Bewegung, die aus Translationen und Rotationen besteht, die Menge der Hindernispolyeder bzw. -polygone berechnet werden, mit denen das bewegte Polyeder bzw. Polygon während der Bewegung kollidiert.

Auf Verfahren zur Kollisionserkennung bei allgemeinen Bewegungen wurde an dieser Stelle nicht eingegangen. Numerische Verfahren zur Lösung dieses Problems finden sich in [Sch94] und [Si94].

2.4 Bewegungsplanungsalgorithmen

In der Folge werden die verschiedenen Kategorien von geometrischen Bewegungsplanungsalgorithmen im Hinblick auf ihre Einsetzbarkeit zusammen mit der heuristisch inkrementellen Bewegungsplanungsstrategie betrachtet.

Grundsätzlich kann man bei Bewegungsplanungsalgorithmen vollständige Algorithmen – Algorithmen, die entscheiden können, ob eine Lösung für ein Bewegungsplanungsproblem existiert – und Heuristiken – Algorithmen, die nicht unbedingt eine Lösung für ein Bewegungsplanungsproblem berechnen können, falls eine existiert – unterscheiden. Vollständige Algorithmen terminieren immer, während Heuristiken u.U. auch nicht terminieren können. Heuristiken, die nicht immer terminieren, sind für den Einsatz zusammen mit der heuristisch inkrementellen Bewegungsplanungsstrategie problematisch, da nicht garantiert werden kann, daß die heuristisch inkrementelle Bewegungsplanungsstrategie immer dann terminiert, wenn die Heuristik angesetzt auf ein Bewegungsplanungsproblem terminiert.

Terminiert eine Heuristik, kann jedoch weder einen Weg berechnen noch entscheiden, daß keiner existiert, hat die heuristisch inkrementelle Bewegungsplanungsstrategie keinen Anhaltspunkt zur Modifikation der Szene in der nächsten Iteration. Die Modifikation kann dadurch vorgenommen werden, daß zufällig ein Hindernis in die Szene der nächsten Iteration aufgenommen wird oder ein Hindernis, das den minimalen Sicherheitsabstand auf dem zuletzt berechneten Weg hatte.

Durch diese Vorgehensweise wird sichergestellt, daß die heuristisch inkrementelle Bewegungsplanungsstrategie genau dann terminiert, wenn der benutzte Bewegungsplanungsalgorithmus für alle Bewegungsplanungsprobleme terminiert.

2.4.1 Klassische Bewegungsplanungsalgorithmen

Verfahren zur klassischen Bewegungsplanung berechnen für ein Polyeder eine Bewegung von einer Start- zu einer Endposition, so daß diese kollisionsfrei bzgl. der betrachteten Hindernispolyeder verläuft, unter Ausnutzung der Bewegungsmöglichkeiten der Polyeder. Das zu lösende Problem kann formal gefaßt werden durch die folgende Problemstellung:

gegeben:

P_{bewegt}	bewegtes Polyeder
H	Menge der Hindernispolyeder
p_{start}	Startposition des bewegten Polyeders
p_{end}	Endposition des bewegten Polyeders

gesucht:

Gibt es eine kollisionsfreie Bewegung für P_{bewegt} von p_{start} nach p_{end}?

Die Laufzeit fast aller Bewegungsplanungsalgorithmen hängt von der Beschreibungskomplexität der betrachteten Szene ab. Dies geschieht durch die Berechnung von Konfigurationsräumen, deren Aussehen von dem bewegten Objekt und den Hindernissen in der Szene abhängt. Eingeführt von [Ud77] und [LPW79] wurde der Konfigurationsraumansatz von Schwartz und Sharir in [SS83a], [SS83b], [SS83c] und [SS84] sowie von Sharir und Ariel-Sheffi in [SA84] weiterentwickelt und insbesondere gezeigt, daß mit Hilfe dieses Ansatzes das allgemeine Bewegungsplanungsproblem gelöst werden kann. Darüberhinaus wurden weitere Algorithmen für Bewegungsplanungsprobleme basierend auf dem Konfigurationsraumansatz entwickelt, wie z.B. von Kedem und Sharir [KS88].

Für alle diese Algorithmen ist es von Vorteil, wenn die Anzahl und damit auch die Komplexität der das Aussehen des Konfigurationsraumes bestimmenden Hindernisse reduziert wird. Dadurch kann i.a. schneller eine Lösung für das gegebene Bewegungsplanungsproblem gefunden werden.

2.4.2 Bewegungsplanungsalgorithmen bei mehreren bewegten Objekten

Bewegungsplanungsalgorithmen mit mehreren bewegten Objekten stellen eine Erweiterung des Konzeptes der klassischen Bewegungsplanung dar und bedürfen des Einsatzes spezieller Algorithmen. Das Bewegungsplanungsproblem bei mehreren bewegten Objekten läßt sich folgendermaßen formal darstellen:

gegeben:

P_{bewegt}	Menge bewegter Polyeder
H	Menge der Hindernispolyeder
$\{p_{start}\}$	Menge von Startpositionen p^i_{start} der bewegten Polyeder $P_i \in P_{bewegt}$
$\{p_{end}\}$	Menge von Endpositionen p^i_{end} der bewegten Polyeder $P_i \in P_{bewegt}$

gesucht:

Gibt es kollisionsfreie Bewegungen für die Polyeder P_i aus P_{bewegt} von p^i_{start} nach p^i_{end}?

Algorithmen zur Lösung spezieller Instanzen dieser Problemklasse wurden von z.B. Erdmann und Lozano-Perez [ELP87] entwickelt. Eine Übersicht darüber wird in [La91] gegeben.

Auch mit diesen Algorithmen ist die heuristisch inkrementelle Vorgehensweise möglich, da Kollisionen zwischen den bewegten Objekten durch den Bewegungsplanungsalgorithmus ausgeschlossen werden. Die Kollisionserkennung für die bewegten Objekte bzgl. der festen Hindernisse kann dadurch für jedes bewegte Objekt einzeln durchgeführt werden. Die gesamte Menge der dabei gefundenen störenden Hindernisse wird in die nächste Iteration übernommen.

Der Algorithmus von Erdmann und Lozano-Perez [ELP87] ist für die heuristisch inkrementelle Bewegungsplanung besonders geeignet, da er auf dem Konfigurationsraumansatz beruht.

Erdmann und Lozano-Perez weisen den zu bewegenden Objekten Prioritäten zu und berechnen sukzessive für die zu bewegenden Objekte die zugehörigen Konfigurationsräume mit zusätzlicher zeitlicher Dimension. Die Wege der zuvor geplanten Objekte werden als Hindernisse in den Konfigurationsraum eingefügt. In dem entstehenden Konfigurationsraum wird ein Weg für das zu bewegende Objekt berechnet.

2.4.3 Bewegungsplanung in dynamischen Umgebungen

Während bei der Bewegungsplanung mit mehreren bewegten Objekten Bewegungen für mehr als ein zu bewegendes Objekt berechnet werden müssen, liegt der Schwerpunkt bei der Bewegungsplanung in dynamischen Umgebungen auf sich bewegenden Hindernissen, die bekannte Wege durchlaufen. Dabei kommt zum ersten Mal der Begriff der Geschwindigkeit der Objekte in die Betrachtung, da diese eine entscheidende Bedeutung für die Bewegungsplanung in dynamischen Umgebungen hat.

Für die heuristisch inkrementelle Vorgehensweise verändert sich dadurch insbesondere die Kollisionserkennung, die bisher nur für ein sich bewegendes Objekt ausgelegt war. Sie muß für die Betrachtung dynamischer Umgebungen modifiziert werden. Sie läßt sich auf die Betrachtung der Kollisionen zweier sich bewegender Objekte zurückführen. Diese beiden Bewegungen lassen sich zu einer Relativbewegung zusammenfassen, so daß wieder der Fall der klassischen Kollisionsrechnung vorliegt, bei dem ein bewegtes Objekt und ein festes Hindernis betrachtet werden. Das Problem besteht in der sich ergebenden Relativbewegung, die aus den Überlagerung der Bewegungen der beiden Körper entsteht.
Haben beide Objekte unterschiedliche Geschwindigkeiten, so ergeben sich aus der Überlagerungen aller möglichen Bewegungsarten, insbesondere auch reiner Translationen bzw. Rotationen, allgemeine Relativbewegungen. Dabei kann nur für den Fall gleicher Geschwindigkeiten beider Objekte bei rein translatorischen Bewegungen diese immer zu einer Relativbewegung für eines der beiden Objekte zusammengefaßt werden, die mit der klassischen Kollisionsrechnung effizient behandelt werden kann. Für alle anderen Fälle muß i.a. bei der benötigten Kollisionsrechnung auf numerische Verfahren zurückgegriffen werden.

Die Benutzung der heuristisch inkrementellen Vorgehensweise ist somit auch zusammen mit Bewegungsplanungsalgorithmen in dynamisch sich verändernden Umgebungen möglich, wobei die Kollisionsrechnung zur Verifikation der Kollisionsfreiheit bzw. zur Berechnung der die

Bewegung störenden Hindernisse komplexer wird. Die Vorteile der heuristisch inkrementellen Vorgehensweise bleiben jedoch bestehen.

Ein Beispiel für einen Algorithmus ist der von Kant und Zucker [KZ86]. Er zerlegt das Gesamtproblem in zwei Teilprobleme. In der ersten Phase wird unabhängig von den bewegten Hindernissen ein Weg für das zu bewegende Objekt mit Hilfe der klassischen Bewegungsplanung berechnet. In einer zweiten Phase muß diese Weg mit denen der bewegten Hindernisse zeitlich koordiniert werden. Dazu wird ein 2-dimensionaler Konfigurationsraum in der Länge der zurückgelegten Wegstrecke und der Zeit berechnet. Freie Positionen sind solche Punkte, bei denen das zu bewende Objekt die zugehörige Weglänge zurückgelegt hat und auf der Endposition nicht mit einem bewegten Hindernisse kollidiert. Dadurch ergeben sich Konfigurationsraumhindernisse. In diesem Konfigurationsraum kann nun ein Weg berechnet werden, aus dem ein Geschwindigkeitsprofil für das zu bewegende Objekt auf dem bereits berechneten Weg abgeleitet wird. In diesem Fall kann die heuristisch inkrementelle Bewegungsplanungsstrategie problemlos in der ersten Phase und ausgestattet mit der entsprechenden Kollisionserkennung auch in der zweiten Phase angewandt werden.

Ein umfassender Überblick über die Bewegungsplanungsalgorithmen in sich dynamisch verändernden Umgebungen wird in [Fu91] und [La91] gegeben.

2.4.4 Zusammenfassung

Grundsätzlich kann die heuristisch inkrementelle Vorgehensweise mit allen Bewegungsplanungsalgorithmen eingesetzt werden. Dabei ist diese Strategie nicht auf die klassische Bewegungsplanung beschränkt, sondern kann auch mit Algorithmen für die Planung von Bewegungen mehrerer Objekte oder in dynamisch sich verändernden Umgebungen benutzt werden.

Besondere Stärken der Vorgehensweise ergeben sich beim gleichzeitigen Einsatz von Bewegungsplanungsalgorithmen, die auf dem von Udupa [Ud77] und Lozano-Perez und Wesley [LPW79] eingeführten Konfigurationsraumansatz basieren, da die Laufzeit zur Berechnung der Konfigurationsräume direkt von der Komplexität der betrachteten Szene abhängt.

In diesen Fällen wirken sich zwei Faktoren positiv auf die Gesamtlaufzeit aus:

1. Die Berechnung von Konfigurationsräumen zu weniger komplexen Szenen ist einfacher als die zu komplexeren Szenen.

2. Der in der vorangegangenen Iteration berechnete Konfigurationsraum kann wiederverwendet und eine Verfeinerung um die neu hinzugekommenen Hindernisse durchgeführt werden.

Insbesondere der letztgenannte Vorteil geht bei Bewegungsplanungsalgorithmen verloren, die den Konfigurationsraum nicht explizit berechnen, wie z.B. [Sch92a]. Jedoch sollte für alle Algorithmen die Berechnung von Lösungen in einfachen Szenarien gegenüber komplexeren Szenarien schneller sein, so daß durch den Einsatz der heuristisch inkrementellen Vorgehensweise die praktische Laufzeit der Algorithmen verringert werden kann.

2.5 Eigenschaften, theoretisches Laufzeitverhalten und Variationen

2.5.1 Eigenschaften

Betrachtet man unter den Bewegungsplanungsalgorithmen die Klasse der terminierenden Algorithmen, und dabei eine spezielle Instanz BPA, so sucht man zur Lösung eines Bewegungsplanungsproblems B eine kollisionsfreie Bewegung $l_{BPA}(B)$ für das bewegte Objekt von der Startposition zur Endposition aus der Menge aller möglichen kollisionsfreien Bewegungen $L(B)$. Dazu setzt man den Algorithmus BPA auf das Bewegungsplanungsproblem an.
I.a. gibt es aber auch ein Bewegungsplanungsproblem $B' \subset B$, das nur eine echte Teilmenge der Hindernisse von B enthält, dessen Lösung $l_{BPA}(B')$ eine Lösung für B ist, d.h. $l_{BPA}(B') \in L(B)$ und mit Hilfe von BPA schneller berechnet werden kann als $l_{BPA}(B)$. Zur schnellen Lösung von B ergibt sich somit das Suchproblem

$$B_{opt} = \min_{B' \subseteq B} \{\text{Laufzeit}_{BPA}(B') \mid l_{BPA}(B') \in L(B)\}$$

Die heuristisch inkrementelle Bewegungsplanungsstrategie stellt dabei eine einfache heuristische Vorgehensweise bei der Suche nach B_{opt} dar. Sie berechnet für eine Folge

$$B_0 \subset B_1 \subset \ldots \subset B_n$$

Lösungen $l_{BPA}(B_i)$ und testet mit Hilfe der Kollisionserkennung, ob $l_{BPA}(B_i) \in L(B)$ ist. Findet sie ein Bewegungsplanungsproblem $B_j \subset B$ mit $l_{BPA}(B_j) \in L(B)$, terminiert das Verfahren. Da in jeder Iteration mindestens ein Hindernis gegenüber der vorherigen Iteration hinzukommt, wird nach höchstens Anzahl der Hindernisse vielen Iterationen das gestellte Bewegungsplanungsproblem untersucht. Dadurch ist sichergestellt, daß die heuristisch inkrementelle Bewegungsplanungsstrategie genau dann terminiert, wenn BPA angesetzt auf ein Bewgungsplanungsproblem immer terminiert.

Dadurch werden folgende Aussagen bewiesen:

Satz 2.12:

Gegeben sei ein Bewegungsplanungsalgorithmus BPA. Sei L_{BPA} die Menge der mit Hilfe von BPA lösbaren Probleme und $L_{BPA}^{inkrementell}$ die Menge der mit Hilfe der heuristisch inkrementellen Bewegungsplanungsstrategie und BPA lösbaren Probleme. Dann gilt für die Anwendung der heuristisch inkrementellen Bewegungsplanung auf BPA:

1. *Ist BPA ein vollständiger Algorithmus, bleibt der Algorithmus bei Anwendung der heuristisch inkrementellen Vorgehensweise vollständig.*
2. *Terminiert BPA für alle Bewegungsplanungsprobleme, terminiert der Algorithmus bei Anwendung der heuristisch inkrementellen Vorgehensweise und es gilt:*

$$L_{BPA} \subseteq L_{BPA}^{inkrementell}$$

Bemerkung 2.13:

1. *Die erste Aussage zeigt insbesondere, daß auch das allgemeine Bewegungsplanungsproblem mit Hilfe der heuristisch inkrementellen Vorgehensweise gelöst werden kann.*
2. *Die zweite Aussage ist insbesondere für heuristische Bewegungsplanungsalgorithmen wichtig, da die Menge der durch die Heuristik lösbaren Bewegungsplanungsprobleme durch die heuristisch inkrementelle Vorgehensweise nicht eingeschränkt, sondern höchstens erweitert wird, wenn die Heuristik für alle Bewegungsplanungsprobleme terminiert.*

2.5.2 Qualitative Wege

Es gibt auch Bewegungsplanungsalgorithmen, die versuchen, nicht nur das Bewegungsplanungsproblem zu lösen, sondern auch dabei qualitative Wege, z.B. kürzeste, sicherste oder zeitminimale, zu berechnen. Für diese drei Fälle werden nun die Auswirkungen der Anwendung der heuristisch inkrementellen Bewegungsplanung diskutiert.

2.5.2.1 Kürzeste Wege

Berechnet der Bewegungsplanungsalgorithmus kürzeste Wege, dann berechnet auch die heuristisch inkrementelle Bewegungsplanung mit diesem Bewegungsplanungsalgorithmus einen kürzesten Weg.

Es ist offensichtlich, daß der kürzeste Weg in dem gestellten Bewegungsplanungsproblem B eine Lösung des Bewegungsplanungsproblems jeder Teilszene $B_i \subset B$ ist. Findet BPA umgekehrt in $B_i \subset B$ einen kürzesten Weg, der eine Lösung für das gestellte Bewegungsplanungsproblem ist, so ist dieser höchstens genausolang wie der kürzeste Weg in B. D.h. der mit Hilfe der heuristisch inkrementellen Bewegungsplanung gefundene Weg ist ein kürzester Weg für das gestellte Bewegungsplanungsproblem. Das zeigt folgenden Satz:

Satz 2.14:

Sei BPA ein Bewegungsplanungsalgorithmus zur Berechnung kürzester Wege, der bei allen Bewegungsplanungsproblemen terminiert, und B ein Bewegungsplanungsproblem. Findet BPA angesetzt auf B einen kürzesten Weg, findet die heuristisch inkrementelle Bewegungsplanungsstrategie mit BPA angesetzt auf B ebenfalls einen kürzesten Weg.

2.5.2.2 Sicherste Wege

Im Gegensatz zur Berechnung kürzester Wege kann bei der Berechnung sicherster Wege beim Einsatz der heuristisch inkrementellen Bewegungsplanungsstrategie nicht mehr garantiert werden, daß die Lösung ein sicherster Weg für das gestellte Problem ist. Das kann dadurch gezeigt werden, daß zu jedem Bewegungsplanungsproblem B, daß mehr als eine Lösung besitzt, durch Hinzufügen eines Hindernisses ein Bewegungsplanungsproblem B' generiert werden kann, für das die sicherste Lösung von B eine kollisionsfreie, aber nicht die sicherste Lösung von B' ist. Das kommt daher, daß die heuristisch inkrementelle Bewegungsstrategie die berechneten Wege zwar auf Kollisionsfreiheit, nicht aber auf deren minimalen Sicherheitsabstand zu den nicht in der Bewegungsplanung betrachteten Hindernissen prüft. Wie in [Sch94] jedoch gezeigt wird, ist die Berechnung des minimalen Sicherheitsabstandes von einem bewegten Objekt zu einem Hindernis im Rahmen der Kollisionserkennung möglich.

Dabei kann der sicherste Weg in einem Bewegungsplanungsproblem B_i in einem Bewegungsplanungsproblem B_j mit $B_i \subset B_j$ nur dann verändert werden, wenn in $B_j \setminus B_i$ ein Hindernis enthalten ist, dessen Sicherheitsabstand zu dem in B_i berechneten sichersten Weg kleiner ist als der minimale Sicherheitsabstand zu den in B_i betrachteten Hindernissen. Beendet man nun die heuristische Suche nicht mit der Berechnung eines kollisionsfreien Weges für das gestellte Problem, sondern erst, wenn kein Hindernis in B einen geringeren Sicherheitsabstand zu dem bewegten Objekt hat als der minimale Sicherheitsabstand des Weges in B_i, so berechnet die heuristisch inkrementelle Bewegungsplanungsstrategie einen sichersten Weg.

Daher kann die heuristisch inkrementelle Bewegungsplanungsstrategie zur Berechnung sicherster Wege erweitert werden. Der modifizierte Algorithmus lautet dann:

H_{nicht}	Menge der aktuell **nicht** betrachteten Hindernisse
$H_{aktuell}$	Menge der aktuell betrachteten Hindernisse
O	bewegtes Objekt
p_{start}	Startposition von O
p_{end}	Zielposition von O
w	berechneter kollisionsfreier Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}
$d_{aktuell}$	minimaler Sicherheitsabstand von w bzgl. $H_{aktuell}$
d_{nicht}	minimaler Sicherheitsabstand von w bzgl. H_{nicht}
d_h	minimaler Sicherheitsabstand von w bzgl. h

$H \leftarrow$ Menge aller Hindernisse
$H_{aktuell} \leftarrow \emptyset$
$w \leftarrow$ beliebiger Weg von p_{start} nach p_{end}
while (w existiert) und ((w nicht kollisionsfrei bzgl. H) oder ($d_{aktuell} > d_{nicht}$)) **do**
    forall $h \in H$ **do**
        if ((O kollidiert mit h auf w) oder ($d_{aktuell} > d_h$)) **then**
            $H_{aktuell} \leftarrow H_{aktuell} \cup h$
            $H_{nicht} \leftarrow H_{nicht} \setminus \{h\}$
    if ($\exists$ kollisionsfreier Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}) **then**
        $w \leftarrow$ kollisionsfreier Weg für O bzgl. $H_{aktuell}$ von p_{start} nach p_{end}
    else
        $w \leftarrow$ existiert nicht

Dadurch wird folgende Aussage gezeigt:

Satz 2.15 :

Sei BPA ein Bewegungsplanungsalgorithmus zur Berechnung sicherster Wege, der bei allen Bewegungsplanungsproblemen terminiert, und B ein Bewegungsplanungsproblem. Findet BPA angesetzt auf B einen sichersten Weg, dann findet die heuristisch inkrementelle Bewegungsplanungsstrategie mit BPA angesetzt auf B ebenfalls einen sichersten Weg.

2.5.2.3 Zeitminimale Wege

Im Zusammenhang mit dynamischen Umgebungen, d.h. in Szenen, in denen neben festen Hindernissen auch bewegte Hindernisse enthalten sind, die bekannte Wege verfolgen (vgl. [Fu91]), werden zeitminimale Wege berechnet. Da durch das Hinzufügen von Hindernissen in eine Szene der Weg für das bewegte Objekt durch die Hindernisse höchstens zusätzlich gestört werden kann, wird der zeitminimale Weg nicht verkürzt. D.h. ein zeitminimaler Weg in einem Teilproblem B_i des gestellten Bewegungsplanungsproblems B, der kollisionsfrei ist bzgl. aller Hindernisse in B, ist auch ein zeitminimaler Weg in B. Dadurch ist folgende Aussage gezeigt:

Satz 2.16 :

Sei BPA ein Bewegungsplanungsalgorithmus zur Berechnung zeitminimaler Wege, der bei allen Bewegungsplanungsproblemen terminiert, und B ein Bewegungsplanungsproblem. Findet BPA angesetzt auf B einen zeitminimalen Weg, dann findet die heuristisch inkrementelle Bewegungsplanungsstrategie mit BPA angesetzt auf B ebenfalls einen zeitminimalen Weg.

2.5.3 Theoretisches Laufzeitverhalten

Es ist klar, daß im Laufzeitverhalten kein Gewinn gegenüber der klassischen Vorgehensweise zu erzielen ist, sondern im Gegenteil eine Verschlechterung in der theoretischen Laufzeit in Kauf genommen werden muß. Grundsätzlich basiert der praktische Laufzeitgewinn auf zwei Trade-Offs von Laufzeiten.

Der erste Trade-Off besteht zwischen der Laufzeit des Bewegungsplanungsalgorithmus für ein gegebenes Bewegungsplanungsproblem und der Laufzeit für die Kollisionserkennung bzgl. aller im Bewegungsplanungsalgorithmus nicht betrachteten Hindernisse. Der Gewinn dabei liegt in der Beobachtung, daß die Betrachtung eines Hindernisses in der Bewegungsplanung in der Laufzeit i.a. sehr viel schwerer wiegt als in der Kollisionserkennung; d.h. es ist sinnvoll das betrachtete Bewegungsplanungsproblem so einfach wie möglich zu gestalten und dabei die Kollisionserkennung zu verkomplizieren.

Der zweite Trade-Off ist in der Zahl der zur Lösung des Bewegungsplanungsproblem benötigten Iterationen der heuristisch inkrementellen Vorgehensweise zu suchen. Die Beobachtung dabei ist, daß es im i.a. besser ist eine Serie von einfachen Bewegungsplanungsproblemen zu lösen als ein sehr komplexes. D.h. der Vorteil, daß die zu lösenden Bewegungsplanungsprobleme einfach sind und daher schnell gelöst werden können, muß ausreichen, den Nachteil einer durchzuführenden Kollisionserkennung und mehrerer Iterationen in der Bewegungsplanung zu superkompensieren.

Dies ist – wie die praktische Beobachtung später zeigen wird – der Fall, in einer worst-case Laufzeitbetrachtung kann dieser Aspekt jedoch nicht zum Tragen kommen.

Für die theoretische Laufzeit der heuristisch inkrementellen Strategie ergibt sich:

n	Gesamtkomplexität der Szene
k	Anzahl der Hindernisse in der Szene
k_i	Komplexität des Hindernis i (i=1,...,k)
w_i	Länge des durch BPA berechneten Weges in Anzahl der Translationen und Rotationen
$BPA(n)$	Laufzeit von BPA angesetzt auf ein Bewegungsplanungsproblem der Komplexität n
$Koll(w,n)$	Laufzeit der Kollisionserkennung angesetzt auf eine Szene der Komplexität n und einem berechneten Weg w

$$\sum_{i=0}^{k}\left[\underbrace{\text{Koll}\left(w_k, \sum_{j=i+1}^{k} k_j\right)^2}_{\text{Laufzeit Kollisionserkennung}} + \underbrace{\text{BPA}\left(\sum_{j=1}^{i} k_j\right)}_{\text{Laufzeit Bewegungsplanungsalgorithmus}}\right]$$

$$= \quad O(k \cdot (\text{Koll}(w_{max}, n)) + BPA(n))$$

wobei w_{max} die Länge des längsten durch BPA berechneten Weges in Anzahl der Translationen und Rotationen ist.

Dies zeigt folgende Aussage:

Satz 2.17 :

Gegeben sei ein Bewegungsplanungsproblem mit k Hindernissen, einer Gesamtkomplexität von n und der Laufzeit des Bewegungsplanungsalgorithmus von $BPA(n)$. Terminiert der Bewegungsplanungsalgorithmus für alle Eingaben, dann kann das Bewegungsplanungsproblem mit Hilfe der heuristisch inkrementellen Vorgehensweise in Zeit

$$O(k \cdot (\text{Koll}(w_{max}, n)) + BPA(n))$$

gelöst werden, wobei $Koll(w_{max}, n)$ die Laufzeit der Kollisionserkennung für den längsten von BPA in dem Bewegungsplanungsproblem berechneten Weg ist.

2.5.4 Variationen der Heuristik

In der Berücksichtigung von Hindernissen, mit denen das bewegte Objekt auf einem in einer Iteration der Heuristik berechneten Weg kollidiert, bietet sich Raum für Variationen der Heuristik. Grundsätzlich ist es möglich, entweder alle kollidierenden Hindernisse in der nächsten Iteration zu berücksichtigen oder nur einen Teil davon. Dies hat keinen Einfluß auf die theoretische Gesamtlaufzeit oder die Eigenschaften der Heuristik, jedoch kann in der Praxis die Laufzeit von der Wahl der in die Folgeiteration aufgenommenen, kollidierenden Hindernisse abhängen.

Die Entscheidung, welche Hindernisse aufgenommen werden sollten, hängt von dem Zustand des berechneten Weges ab.
Befindet sich die Heuristik noch in einer Suchphase, d.h. verändert sich der berechnete Weg in der darauffolgenden Iteration nicht lokal sondern global, dann verändert in der Regel das erste Hindernis, das die berechnete Bewegung stört, den Weg in der nächsten Iteration derartig, daß andere als störend berechnete Hindernisse für die weitere Wegeplanung nicht relevant sind. Daher ist es in der Suchphase sinnvoll, nur das erste kollidierende Hindernis in die nachfolgende Iteration aufzunehmen.
Ist jedoch die Grobplanung für die Bewegung abgeschlossen und der berechnete Weg wird durch die störenden Hindernisse nur lokal verändert, so daß alle kollidierenden Hindernisse für die weitere Wegeplanung relevant sind, ist es sinnvoll, alle Hindernisse in die nachfolgende Iteration aufzunehmen.

Die in dieser Beschreibung genannten Kriterien sind jedoch nicht einfach in eine Bewertungsfunktion zu fassen, die die Wahl zwischen den verschiedenen Varianten der Heuristik erlauben würde. In *Anhang A* werden bei den praktischen Laufzeitmessungen die beiden oben beschriebenen Varianten der Heuristik als **minimal inkrementell** und **inkrementell** verwendet.

2.6 Zusammenfassung

In diesem Kapitel wurde das Konzept der heuristisch inkrementellen Bewegungsplanung im $\mathbb{R}^3$ beschrieben. Dabei wurde eine Vorgehensweise entwickelt, die grundsätzlich mit allen Bewegungsplanungsalgorithmen benutzt werden kann. Die Kollisionserkennung wurde für 2- und 3-dimensionale Szenarien dargestellt, wobei die Beschreibung auf den Fall von Translationen und Rotationen beschränkt wurde. Es gibt jedoch auch Kollisionserkennungsalgorithmen, die allgemeine Bewegungen auf Kollisionen testen können. Da jedoch die meisten der bekannten Bewegungsplanungsalgorithmen rein translatorische, rotatorische oder stückweise translatorische bzw. rotatorische Bewegungen berechnen, wurde auf die Beschreibung von Kollisionserkennungsalgorithmen für allgemeine Bewegungen verzichtet.

Mit den im vorigen Abschnitt dargestellten Eigenschaften der heuristisch inkrementellen Bewegungsplanungsstrategie kann gesagt werden, daß es sich um eine allgemeine Strategie zur Beschleunigung von Bewegungsplanungsalgorithmen handelt. In der theoretischen Laufzeit ist erwartungsgemäß keine Verbesserung gegenüber den bekannten Algorithmen möglich, jedoch zeigen sich Vorteile in der praktischen Anwendung, wie durch Laufzeitergebnisse in *Anhang A* veranschaulicht wird.

Kapitel 3

Anwendung der heuristisch inkrementellen Bewegungsplanung im $\mathbb{R}^3$

Nachdem nun im vorhergehenden Kapitel die heuristisch inkrementelle Vorgehensweise allgemein erläutert worden ist, soll in diesem Kapitel eine konkrete Anwendung für diese Strategie beschrieben werden und anhand dieser Anwendung die Wirkungen aufgezeigt werden. Der in der Folge beschriebene klassische Bewegungsplanungsalgorithmus wurde im Rahmen dieser Arbeit im $\mathbb{IGOR}$-System implementiert und um die heuristisch inkrementelle Vorgehensweise erweitert.

Das $\mathbb{IGOR}$-System arbeitet mit 3-dimensionalen Objektdaten, so daß ein Bewegungsplanungsalgorithmus zur Bearbeitung von 3-dimensionalen Objektdaten implementiert wurde. Der gewählte Bewegungsplanungsalgorithmus basiert auf dem Konfigurationsraumansatz und bietet damit auch gute Voraussetzungen zum Test der Wirkung der heuristisch inkrementellen Bewegungsplanung.

Der Algorithmus erlaubt die Planung von planaren, translatorischen Bewegungen für ein zu bewegendes Objekt bei freier Wahl der Bewegungsebene. Dabei sind nichtkonvexe Polyeder als zu bewegendes Objekt und als Hindernisse zulässig.

Bei Benutzung des Konfigurationsraumansatzes lassen sich die Berechnungen in zwei große Teile untergliedern:

1. Berechnung des Konfigurationsraumes
2. Berechnungen im Konfigurationsraum

Die Berechnung des Konfigurationsraumes entspricht einer Transformation des gestellten Bewegungsplanungsproblems. Im Objektraum – dem Raum, in dem die Bewegungen stattfinden – werden Start- und Endposition des zu bewegenden Objektes spezifiziert. Dadurch ist ein Bewegungsplanungsproblem im Objektraum definiert, bei dem ein Objekt bewegt werden soll.

Zu dessen Lösung wird das Problem in einen 2-dimensionalen Konfigurationsraum transformiert, in dem das Problem, ein 3-dimensionales Objekt zu bewegen, umgesetzt wird in ein Problem, einen Punkt zu bewegen.

Die Position eines Objektes im $\mathbb{R}^3$ kann durch die Angabe von sechs Parametern eindeutig bestimmt werden, wobei drei Parameter zur Beschreibung der Position des Referenzpunktes des Objektes und drei Parameter zur Beschreibung der Orientierung des Objektes benötigt werden, d.h. ein Objekt im $\mathbb{R}^3$ hat sechs Freiheitsgrade. In dem zugehörigen Konfigurationsraum entspricht jeder Punkt eindeutig einer Plazierung des Objektes, so daß ein 6-dimensionaler Konfigurationsraum nötig wäre, um alle Positionen des Objektes zu beschreiben.

Der Algorithmus berechnet von dem 6-dimensionalen Konfigurationsraum, der alle Positionen des bewegten Objektes beschreibt, einen 2-dimensionalen Unterraum für feste Werte der übrigen Positionierungsparameter. In der Folge werden nun die einzelnen Teile des Bewegungsplanungsalgorithmus beschrieben, wie er im Rahmen dieser Arbeit innerhalb des $\mathbb{IGOR}$ -Systems realisiert wurde.

3.1 Berechnung des Konfigurationsraumes

Wie bereits in *Kapitel 1* beschrieben, ist bei der Transformation des Problems vom Objektraum in den Konfigurationsraum nicht das gesamte Aussehen des Konfigurationsraumes für die Lösung des Bewegungsplanungsproblems relevant, sondern lediglich die Zusammenhangskomponente des Freiraums, die die Startposition der Bewegung beinhaltet. Andere Zusammenhangskomponenten des Freiraums sind von der Startposition aus nicht erreichbar. Ziel der Berechnung des Konfigurationsraumes ist also nicht die Berechnung des kompletten Aussehens des Konfigurationsraumes, sondern der Zusammenhangskomponente des Konfigurationsraumes, die die Startposition der Bewegungs enthält.

Erste Ansätze und Algorithmen, wie solche Konfigurationsräume für ein bewegtes Objekt explizit berechnet werden können, beschreibt Lozano-Perez in [LP83]. Es wird gezeigt, daß das zu einem bewegten Polyeder O und einem festen Hindernispolyeder H gehörige Konfigurationsraumhindernis ein Polyeder ist, das der Minkowski-Differenz zwischen den beiden Polyedern entspricht. Mit Hilfe der Mengen-Differenz

$$A \ominus B \quad = \quad \{a - b \mid a \in A, b \in B\}$$

ergibt sich das Konfigurationsraumhindernis von O und H bei 3 translatorischen Freiheitsgraden zu

$$CO_O^3(H) = H \ominus O$$

Für die Berechnung dieser Minkowski-Differenz für konvexe Polyeder wird in [La91] ein effizienter Algorithmus von Guibas und Seidel [GS86] angegeben, der Laufzeit $O(|H| + |O| + |CO_O^3(H)|)$ hat, was im worst-case $O(|H| \cdot |O|)$ bedeutet. Für den Fall nichtkonvexer Polyeder wird an gleicher Stelle auf einen Algorithmus von Avnaim und Boissonnat [AB88] verwiesen, der in Zeit $O(|H|^3 \cdot |O|^3 \cdot \log(|H| \cdot |O|))$ arbeitet.

Soll nun ein Konfigurationsraum für zwei translatorische Freiheitsgrade berechnet werden, kann der Schnitt der Minkowski-Differenz mit der Translationsebene gebildet werden, die die Startposition der Bewegung enthält. Die Konfigurationsraumebene ergibt sich dann aus der Ebenengleichung

$$P_{trans} = \{\mathbf{p} \in \mathbb{R}^3 \mid \mathbf{n}^T(\mathbf{p} - \mathbf{p}_{start}) = 0\}$$

und der Minkowski-Differenz

$$H \ominus O$$

als Schnittmenge

$$CO_O^2(H) = P_{trans} \cap (H \ominus O)$$

Nach dieser Beschreibung ergibt sich somit eine kanonische Vorgehensweise für die Berechnung des Konfigurationsraumes. Da die Berechnung des Schnittes mit der Translationsebene in Linearzeit bzgl. der Größe der Minkowski-Differenz erfolgen kann, ergibt sich eine Laufzeit von $O(|H| \cdot |O|)$ bei konvexen und von $O(|H|^3 \cdot |O|^3 \cdot \log(|H| \cdot |O|))$ bei nichtkonvexen Polyedern. Bei dieser Vorgehensweise erfolgt zunächst die Berechnung eines höherdimensionalen Konfigurationsraumes, so daß bei der direkten Berechnung des eingeschränkten Konfigurationsraums ein Laufzeitgewinn zu erwarten ist.

Jedoch zeigt die Vorgehensweise über die Berechnung des 3-dimensionalen Konfigurationsraums, welches Aussehen der zu berechnende 2-dimensionale Konfigurationsraum hat. Die Konfigurationsraumhindernisse des 3-dimensionalen Konfigurationsraumes sind Polyeder und polyhedrale Regionen, so daß die Hindernisse im 2-dimensionalen Konfigurationsraum Polygone und polygonale Regionen sind.

Die Begrenzungen der Konfigurationsraumhindernisse entsprechen Kontaktsituationen zwischen bewegtem Objekt und Hindernis im Objektraum. Daher werden die Kontaktsituationen zwischen zwei Polyedern betrachtet, um die Begrenzung der Konfigurationsraumhindernisse zu berechnen. Dabei werden zuerst lokale Kontaktsituationen zwischen Polyedern betrachtet, bei denen einzelne Polyederteile miteinander in Kontakt kommen. Dadurch wird eine Obermenge von Kontaktpositionen im Konfigurationsraum berechnet, aus denen die Beschreibung der Konfigurationsraumhindernisse berechnet wird.

Bei der Kollisionserkennung ergab sich durch die Vorgabe einer Translation eines bewegten Polyeders, d.h. mit einem Freiheitsgrad, eine punktförmige Berührung. Für den Fall von zwei translatorischen Freiheitsgraden ergeben sich segmentartige Berührungen und bei drei translatorischen Freiheitsgraden Flächen als Berührungen. Mit Hilfe der Kontaktsituationen kann die Beschreibung der Konfigurationsraumhindernisse erfolgen. Bei zwei translatorischen Freiheitsgraden ergibt sich somit eine Menge von lokalen Kontaktsegmenten, aus denen die Beschreibung der Konfigurationsraumhindernisse berechnet werden kann.

Die Berechnung des Konfigurationsraumes bei zwei translatorischen Freiheitsgraden gliedert sich somit in zwei Teile:

1. Berechnung der lokalen Kontaktpositionen

2. Berechnung der Konfigurationsraumhindernisse aus den lokalen Kontaktpositionen

3.1.1 Berechnung der lokalen Kontaktpositionen

Ausgehend von der Szene im Objektraum und der Angabe der Translationsebene wird die Menge aller Kontaktpositionen zwischen bewegtem Polyeder und Hindernispolyeder berechnet. Dabei wird angenommen, daß die Startposition des bewegten Polyeders dem Nullpunkt des Konfigurationsraumes entspricht. Formal läßt sich das Problem spezifizieren als

gegeben:

O	bewegtes Polyeder
H	festes Polyeder
$\mathbf{d}_1, \mathbf{d}_2$	Richtungsvektoren der Translationsebene

gesucht:

$P_K = \{\mathbf{p} = (\alpha, \beta) \in \mathbb{R}^2 \mid \exists\, o \in bound(O),\ h \in bound(H)\ \text{ mit } o + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 = h\}$

Mit Hilfe der Kontaktsituationen zwischen zwei Polyedern wird in der Folge gezeigt, daß die Menge P_K eine Menge von Liniensegmenten ist und in dieser Form effizient berechnet werden kann.

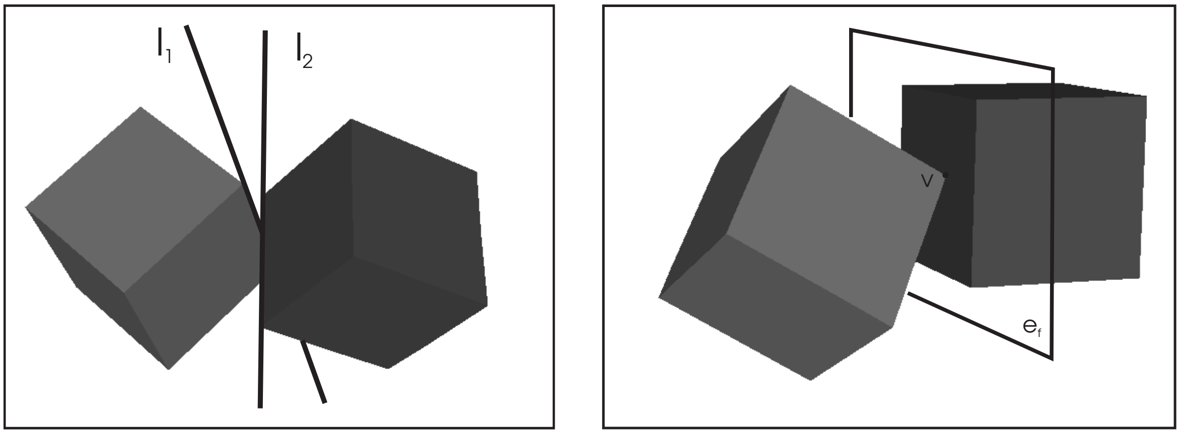

Abbildung 3.1: Kontaktsituationen zwischen Polyedern

Wenn zwei Polyeder miteinander in Kontakt kommen, ist grundsätzlich jeweils eine Fläche des einen und des anderen Polyeders beteiligt. Betrachtet man die Kontaktmöglichkeiten jedoch genauer, ist zu erkennen, daß bei jedem möglichen Kontakt einer der beiden folgenden Fälle vorliegt, wie Abbildung 3.1 verdeutlicht:

- Kante–Kante–Kontakt
- Ecke–Fläche–Kontakt

Satz 3.1 :

Seien P_1 und P_2 zwei beliebige Polyeder. P_1 und P_2 sind genau dann in Kontakt miteinander, wenn

1. $int(P_1) \cap int(P_2) = \emptyset$
2. (a) $\exists\ v \in V_{P_i}, f \in F_{P_{\bar{i}}}$ *mit* $v \in f$
   oder
   (b) $\exists\ e_1 \in E_{P_1}, e_2 \in E_{P_2}$ *mit* $e_1 \cap e_2 \neq \emptyset$

Beweis:
„$\Longleftarrow$ “: klar.

„$\Longrightarrow$ “

1. klar, da sonst zwischen P_1 und P_2 keine Kontakt- sondern eine Schnittsituation vorliegen würde.
2. P_1 in Kontakt mit $P_2 \Rightarrow \exists \mathbf{p}_1 \in \text{bound}(P_1), \mathbf{p}_2 \in \text{bound}(P_2)$ mit $\mathbf{p}_1 = \mathbf{p}_2$. Läge $\mathbf{p}_1$ oder $\mathbf{p}_2$ nicht auf der Polyederhülle, würden sich die beiden Polyeder überlappen und nicht berühren. Sei also $\mathbf{p}_1 \in f_1$ und $\mathbf{p}_2 \in f_2$.
   (a) $f_1 \not\parallel f_2$
      (1) $\mathbf{p}_1 \in int(f_1), \mathbf{p}_2 \in int(f_2)$

      $$\begin{aligned} &\mathbf{p}_i \in int(f_i) \Rightarrow \exists\ \delta_i \text{ mit } Kreis(\mathbf{p}_i, \delta_i) \subset f_i \text{ und} \\ &\qquad\qquad offeneKugel(\mathbf{p}_i, \delta_i) \cap int(P_i) = offeneHalbkugel(\mathbf{p}_i, \delta_i) \\ &\Rightarrow \text{ für } \delta = \min\{\delta_1, \delta_2\} \text{ gilt wegen } f_1 \not\parallel f_2 : \\ &\quad offeneHalbkugel(\mathbf{p}_1, \delta) \cap offeneHalbkugel(\mathbf{p}_2, \delta) \neq \emptyset \\ &\Rightarrow int(P_1) \cap int(P_2) \neq \emptyset \text{ (Widerspruch)} \end{aligned}$$

      (2) $\mathbf{p}_1 \in bound(f_1), \mathbf{p}_2 \in int(f_2)$
      Falls $\exists\ \mathbf{p}' \in int(f_1), \mathbf{p}' \in int(f_2)$ so folgt der Widerspruch aus (1)
         (a) $\mathbf{p}_1 \in V_{P_1} \Rightarrow$ Ecke–Fläche–Kontakt
         (b) $\mathbf{p}_1 \in E_{P_1} \setminus V_{P_1}$

         $$\begin{aligned} \mathbf{p}_2 \in int(f_2) \Rightarrow\ &\exists\ \alpha \in \mathbb{R}, \mathbf{d} = \mathbf{n}_1 \times \mathbf{n}_2 \text{ mit } \mathbf{p}' = \mathbf{p}_2 + \alpha\mathbf{d} \text{ und} \\ &\mathbf{p}' \in bound(f_1) \text{ und } \mathbf{p}' \in bound(f_2) \\ &\Rightarrow \text{Kante–Kante–Kontakt} \\ &\qquad\qquad \textit{oder} \\ &\mathbf{p}' \in V_{P_1} \text{ und } \mathbf{p}' \in int(f_2) \\ &\Rightarrow \text{Ecke–Fläche–Kontakt} \end{aligned}$$

         (vgl. Abbildung 3.2)
      (3) $\mathbf{p}_1 \in int(f_1),\ \mathbf{p}_2 \in bound(f_2)$ analog zu Fall(2)
      (4) $\mathbf{p}_1 \in bound(f_1), \mathbf{p}_2 \in bound(f_2) \Rightarrow$ Kante–Kante–Kontakt

(b) $f_1 \parallel f_2$

(1) $\mathbf{p}_1 \in int(f_1), \mathbf{p}_2 \in int(f_2)$
Dann kann man sich in der von f_1 und f_2 unterstützten Ebene $e(f_1) = e(f_2)$ in eine beliebige Richtung $\mathbf{d}$ innerhalb der Ebene bewegen, bis mit $\mathbf{p}^{'}$ eine Begrenzung von f_1 oder f_2 erreicht wird. O.B.d.A. $\mathbf{p}^{'} \in e_1 \in E_{P_1}$
- $\mathbf{p}^{'} \in V_{P_1} \Rightarrow$ Ecke–Fläche–Kontakt
- Verfolge e_1 bis zu einem Endpunkt v_1
  $v_1 \in f_2 \Rightarrow$ Ecke–Fläche–Kontakt
  $v_1 \notin f_2 \Rightarrow \exists \mathbf{p}^{''} \in e_1$ mit $\mathbf{p}^{''} \in e_2 \in E_{P_2} \Rightarrow$ Kante–Kante–Kontakt

(2) $\mathbf{p}_1 \in int(f_1), \mathbf{p}_2 \in bound(f_2)$ analog zu (1)

(3) $\mathbf{p}_1 \in bound(f_1), \mathbf{p}_2 \in bound(f_2) \Rightarrow$ Kante–Kante–Kontakt

■

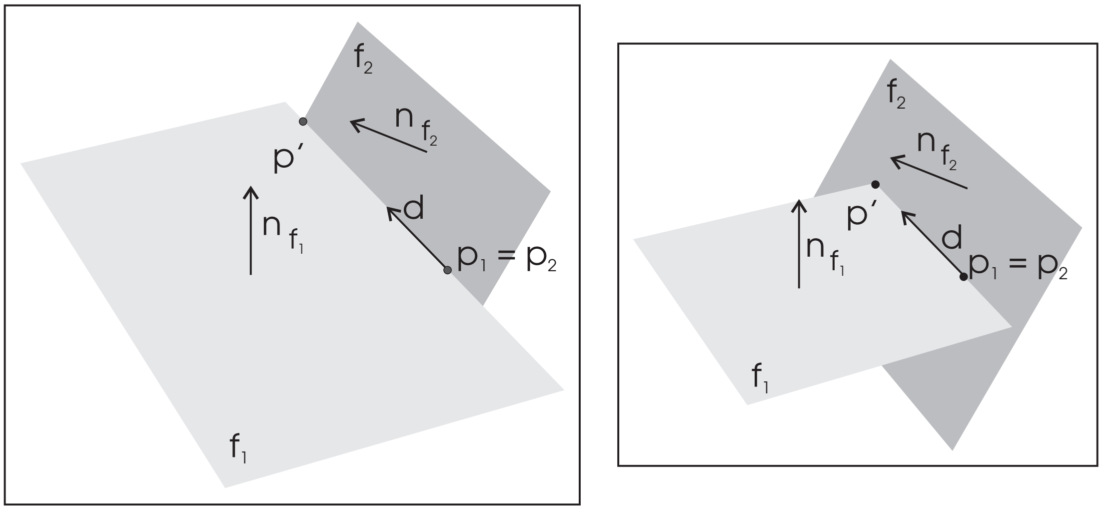

Abbildung 3.2: Kante–Kante–Kontakt und Ecke–Fläche–Kontakt bei kontaktierenden Flächen

Damit ist gezeigt, mit Hilfe welcher lokaler Betrachtungen die Kontaktsituationen zwischen zwei Polyedern berechnet werden können. In der Folge werden nun die zwei Kontaktmöglichkeiten näher betrachtet.

3.1.1.1 Ecke–Fläche–Kontakt

Für die Berechnung der Gesamtmenge aller Kontaktsituationen zwischen zwei Polyedern wird nun folgendes lokales Teilproblem betrachtet:
gegeben:

f	konvexe Fläche eines Polyeders
$\mathbf{v}$	Ecke eines Polyeders
$\mathbf{d}_1, \mathbf{d}_2$	Richtungsvektoren der Translationsebene

gesucht:

$P_K = \{\mathbf{p} = (\alpha, \beta) \in \mathbb{R}^2 \mid \mathbf{v} + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 \in f\}$

Anschaulich betrachtet wird die Parametrisierung des Schnittes der Fläche f mit der Translationsebene, in der die Ecke $\mathbf{v}$ liegt, bzgl. der beiden Richtungsvektoren $\mathbf{d}_1$ und $\mathbf{d}_2$ gesucht. Dazu berechnet man zunächst den Schnitt der Translationsebene $e(\mathbf{v})$ durch den Punkt $\mathbf{v}$ mit der durch die Fläche f unterstützten Ebene $e(f)$. Die durch die Fläche unterstützte Ebene sei gegeben durch

$$e(f) : \mathbf{n}_f^T(\mathbf{x} - \mathbf{v}_0) = 0, \;\; \mathbf{v}_0 \in f$$

Die Translationsebene durch den Eckpunkt $\mathbf{v}$ wird beschrieben durch

$$e(v) = \{\mathbf{p} \mid \mathbf{p} = \mathbf{v} + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2, \;\; \alpha, \beta \in \mathbb{R}\}$$

so daß der Schnitt der beiden Ebenen gegeben ist durch die Gleichung

$$\begin{array}{rrcl} & \mathbf{n}_f^T(\mathbf{v} + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 - \mathbf{v}_0) & = & 0 \\ \Leftrightarrow & \alpha\mathbf{n}_f^T\mathbf{d}_1 + \beta\mathbf{n}_f^T\mathbf{d}_2 & = & \mathbf{n}_f^T\mathbf{v}_0 - \mathbf{n}_f^T\mathbf{v} \end{array}$$

Diese Gleichung parametrisiert eine Gerade im $\mathbb{R}^2$. Eine Degeneration tritt auf, wenn sowohl $\mathbf{n}_f^T\mathbf{d}_1 = 0$ als auch $\mathbf{n}_f^T\mathbf{d}_2 = 0$ ist. Anschaulich bedeutet dies, daß die Translationsebene und die durch die Fläche f unterstützte Ebene parallel liegen. Ist $\mathbf{n}_f^T\mathbf{v}_0 - \mathbf{n}_f^T\mathbf{v} \neq 0$, sind die beiden Ebenen verschieden und es gibt keinen Schnitt; ist $\mathbf{n}_f^T\mathbf{v}_0 - \mathbf{n}_f^T\mathbf{v} = 0$, sind die beiden Ebenen identisch und der Schnitt ist die gesamte Ebene. Betrachtet man sich die Situation anhand der Polyeder, bedeutet dies, daß eine Ecke des einen Polyeders an einer Fläche des anderen Polyeders entlangschleift, ohne in das Innere des anderen Polyeders zu gelangen. Da die Hindernisse in der Bewegungsplanung als offene Mengen modelliert sind, werden durch diese Kontaktsituation zwar halbfreie Positionen beschrieben, jedoch keine freien Positionen von unfreien Positionen im Konfigurationsraum getrennt, so daß diese bei der Berechnung der Konfigurationsraumhindernisse nicht berücksichtigt werden dürfen. D.h. in den degenerierten Fällen werden keine für die Bewegungsplanung relevanten Kontaktsituationen beschrieben. Im nichtdegenerierten Fall beschreibt die Parametrisierung des Schnittes eine Gerade im Konfigurationsraum.

Diese Gerade im Konfigurationsraum wird durch die Beschränkung der Ebene $e(f)$ auf die Fläche f selbst auf ein Liniensegment beschränkt. Dazu betrachtet man sukzessive alle Begrenzungskanten der Fläche und berechnet die Parametrisierung bzgl. $\mathbf{d}_1$ und $\mathbf{d}_2$ des Schnittes der Translationsebene durch $\mathbf{v}$ mit der Begrenzungskante, falls dieser existiert. Zunächst berechnet man für jede Kante den Schnitt der von ihr unterstützten Geraden mit der Translationsebene durch $\mathbf{v}$ mittels

$$\begin{array}{rrcl} & \mathbf{v}_i + \gamma(\mathbf{v}_{i+1} - \mathbf{v}_i) & = & \mathbf{v} + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 \\ \Leftrightarrow & \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 - \gamma(\mathbf{v}_{i+1} - \mathbf{v}_i) & = & \mathbf{v}_i - \mathbf{v} \end{array}$$

Für $\det[\mathbf{v}_{i+1} - \mathbf{v}_i, \mathbf{d}_1, \mathbf{d}_2] = 0 \wedge (\mathbf{d}_1 \times \mathbf{d}_2)^T(\mathbf{v}_i - \mathbf{v}) = 0$ liegt die Gerade in der Translationsebene durch $\mathbf{v}$ und man berechnet direkt die Parametrisierung der Endpunkte der Kante bzgl. $\mathbf{d}_1$ und $\mathbf{d}_2$. Falls die Gerade in einer Parallelebene zur Translationsebene durch $\mathbf{v}$ liegt, ergibt sich kein Schnitt der Translationsebene durch $\mathbf{v}$ mit der Geraden. Im nichtdegenerierten Fall besitzt das Gleichungssystem eine eindeutige Lösung, die die gewünschte Parametrisierung beinhaltet. Ein Schnitt der Begrenzungskante mit der Translationsebene durch $\mathbf{v}$ liegt nur vor, wenn $0 \leq \gamma \leq 1$ gilt. In diesem Fall wird der Punkt (α, β) in die Beschreibung der Geraden im Konfigurationsraum aufgenommen.

Durch die sukzessive Bearbeitung der Begrenzungskanten werden die Parametrisierungen der Schnittpunkte wegen der Konvexität der Fläche in einer zyklischen Sortierung berechnet. Mit Hilfe eines linearen Laufes über die Liste der Schnittpunkte kann das Element mit minimalem gerichtetem Abstand von einem beliebigen Punkt der Geraden berechnet werden. An dieser Stelle wird die zyklische Sortierung aufgebrochen und eine lineare Sortierung erreicht. Das minimale und das maximale Element dieser Sortierung definieren die Endpunkte eines Liniensegmentes im Konfigurationsraum, das der Parametrisierung bzgl. $\mathbf{d}_1$ und $\mathbf{d}_2$ der Schnittpositionen der Ecke $\mathbf{v}$ mit der Fläche f entspricht.

Insgesamt erhält man durch die Betrachtung eines **Ecke–Fläche–Kontaktes** kein oder ein Liniensegment im Konfigurationsraum, das zur Beschreibung der Begrenzung eines Konfigurationsraumhindernisses beitragen und in Zeit $O(|f|)$ berechnet werden kann.

Dies läßt sich zusammenfassen in folgendem Lemma:

Lemma 3.2 :

> *Für eine Ecke* $\mathbf{v}$ *und eine konvexe Fläche* f *im* $\mathbb{R}^3$ *, sowie eine Translationsebene* e*, in der sich die Ecke bewegt, kann in Zeit* $O(|f|)$ *die Parametrisierung der Schnittpunkte der Ecke mit der Fläche bzgl. der Translationsebene in Form eines Liniensegmentes berechnet werden.*

3.1.1.2 Kante–Kante–Kontakt

Neben dem bereits beschriebenen Ecke–Fläche–Kontakt muß als zweites und letztes lokales Teilproblem der Kante–Kante–Kontakt betrachtet werden. Das Problem läßt sich formal spezifizieren als

gegeben:

e_1	Kante des bewegten Polyeders
e_2	Kante des festen Polyeders
$\mathbf{d}_1, \mathbf{d}_2$	Richtungsvektoren der Translationsebene für e_1

gesucht:

$$P_K = \{\mathbf{p} = (\alpha, \beta) \in \mathbb{R}^2 \mid \exists \mathbf{p}_1 \in e_1, \mathbf{p}_2 \in e_2 \text{ mit } \mathbf{p}_1 + \alpha \mathbf{d}_1 + \beta \mathbf{d}_2 = \mathbf{p}_2\}$$

Anschaulich betrachtet ist analog zum Ecke–Fläche–Kontakt die Parametrisierung des Schnittes der Kante e_1 mit der Kante e_2 bzgl. den Richtungsvektoren der Translationsebene $\mathbf{d}_1$ und $\mathbf{d}_2$ gesucht. Dazu berechnet man zunächst die Parametrisierung des Schnittes der durch die beiden Kanten e_1 und e_2 unterstützten Geraden l_1 und l_2. Seien $\mathbf{a}$ und $\mathbf{b}$ die beiden Endpunkte von e_1 und $\mathbf{c}$ und $\mathbf{d}$ die beiden Endpunkte von e_2. Die beiden Geraden ergeben sich dann zu

$$\begin{aligned} l_1 &= \{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{x} = \mathbf{a} + \gamma(\mathbf{b} - \mathbf{a}),\ \gamma \in \mathbb{R}\} \\ l_2 &= \{\mathbf{y} \in \mathbb{R}^3 \mid \mathbf{y} = \mathbf{c} + \delta(\mathbf{d} - \mathbf{c}),\ \delta \in \mathbb{R}\} \end{aligned}$$

Die Lösung des Schnittproblems zwischen den beiden Geraden entspricht damit der Lösung des Gleichungssystems

$$\begin{array}{rrcl} & (l_1 + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2) \cap l_2 & \neq & \emptyset \\ \Leftrightarrow & \mathbf{a} + \gamma(\mathbf{b} - \mathbf{a}) + \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 & = & \mathbf{c} + \delta(\mathbf{d} - \mathbf{c}) \\ \Leftrightarrow & \alpha\mathbf{d}_1 + \beta\mathbf{d}_2 + \gamma(\mathbf{b} - \mathbf{a}) - \delta(\mathbf{d} - \mathbf{c}) & = & \mathbf{c} - \mathbf{a} \end{array}$$

Als Degeneration des Gleichungssystems kann auftreten, daß die beiden Geraden parallel zu der durch $\mathbf{d}_1$ und $\mathbf{d}_2$ aufgespannten Ebene liegen. In diesem Fall hat das Gleichungssystem keine Lösung, falls die beiden Geraden in verschiedenen Parallelebenen zu der Translationsebene liegen, die komplette Konfigurationsraumebene, falls die beiden Geraden in derselben Parallelebenen zu der Translationsebene liegen und selbst nicht parallel sind oder eine Gerade von Lösungen im Konfigurationsraum, falls die beiden Geraden in derselben Parallelebene zur Translationsebene liegen und selbst parallel sind.
Im ersten Fall ist ein Schnitt zwischen den beiden Geraden und somit auch zwischen den beiden Kanten ausgeschlossen; im zweiten Fall beschreibt die durch die Einschränkung der beiden Geraden auf die Kanten sich ergebende Fläche von Kontaktsituationen eine Menge von halbfreien Positionen des bewegten Polyeders, die keine freien von unfreien Positionen trennt und somit nicht zur Beschreibung eines Konfigurationsraumhindernisses beiträgt; der dritte Fall entspricht einem degenierten zweiten Fall, in dem die Fläche zu einem Liniensegment degeneriert. In allen Fällen entstehen keine Kontaktsituationen, die zur Beschreibung von Konfigurationsraumhindernissen beitragen, so daß im Falle einer Degeneration keine Berechnung von Kontaktsegmenten erfolgt. Die Degeneration läßt sich erkennen durch den Test $(\mathbf{d}_1 \times \mathbf{d}_2)(\mathbf{b} - \mathbf{a}) = 0 \wedge (\mathbf{d}_1 \times \mathbf{d}_2)(\mathbf{d} - \mathbf{c}) = 0$.

Im nichtdegenerierten Fall ergibt sich durch die einfache Unterbestimmtheit des Gleichungssystems eine Gerade von Kontaktsituationen im Konfigurationsraum. Schränkt man nun die beiden Geraden auf die Kanten ein, d.h. läßt man nur Lösungen zu, für die $0 \leq \gamma \leq 1$ und $0 \leq \delta \leq 1$ gilt, erhält man ein Liniensegment von Kontaktsituationen im Konfigurationsraum.

Um dieses zu berechnen, betrachtet man zunächst die Einschränkung der Lösung des Gleichungssystems auf jeweils eine der beiden Kanten. Dazu berechnet man für die Endpunkte der Kante die eindeutig bestimmte Lösung des Gleichungssystems und erhält damit für jede der beiden Kanten ein Liniensegment im Konfigurationsraum. Beide Liniensegmente liegen auf einer Geraden und der überlappende Teil der beiden Segmente entspricht den Kontaktsituationen der beiden Kanten. Durch die Beschränkung der Berechnung auf die Endpunkte der Kanten muß zur Berechnung der Kontaktsituationen nur eine konstante Anzahl von Gleichungssytemen mit drei Unbekannten gelöst werden, so daß das entstehende Liniensegment von Kontaktsituationen im Konfigurationsraum in Zeit O(1) berechnet werden kann.
Somit läßt sich das Ergebnis zusammenfassen in folgendem Lemma:

Lemma 3.3:

> *Für zwei Kanten e_1 und e_2 im $\mathbb{R}^3$ und einer Translationsebene e, parallel zu der sich e_1 bewegt, kann in Zeit O(1) die Parametrisierung der Schnittpunkte zwischen den beiden Kanten bzgl. der Richtungsvektoren der Translationsebene in Form eines Liniensegmentes berechnet werden.*

Mit Hilfe der beiden elementaren Berechnungen von Ecke–Fläche– und Kante–Kante–Kontakten ergibt sich folgender Algorithmus zur Berechnung aller möglichen lokalen Kontaktsituationen zwischen Polyedern. Dabei ist für die Berechnung der Ecke–Fläche–Kontakte zwischen den Ecken des festen Polyeders und den Flächen des bewegten Polyeders zu beachten, daß dabei die Richtung der Richtungsvektoren umgekehrt werden muß, um die korrekten Kontaktsegmente im Konfigurationsraum zu berechnen.

F_i	Menge aller Flächen des Polyeders i	$i = 1, 2$
E_i	Menge aller Kanten des Polyeders i	
V_i	Menge aller Ecken des Polyeders i	
f_i	Fläche des Polyeders i	
e_i	Kante des Polyeders i	
v_i	Ecke des Polyeders i	
K	Menge der Kontaktsegmente	
$\mathbf{d}_1, \mathbf{d}_2$	Richtungsvektoren der Translationsebene	

$K \leftarrow \emptyset$
forall $\mathbf{v}_1 \in V_1$ **do**
  forall $f_2 \in F_2$ **do**
    $K \leftarrow K \cup$ Ecke–Fläche–Kontakt$(\mathbf{v}_1, f_2, \mathbf{d}_1, \mathbf{d}_2)$
forall $\mathbf{v}_2 \in V_2$ **do**
  forall $f_1 \in F_1$ **do**
    $K \leftarrow K \cup$ Ecke–Fläche–Kontakt$(\mathbf{v}_2, f_1, -\mathbf{d}_1, -\mathbf{d}_2)$
forall $e_1 \in E_1$ **do**
  forall $e_2 \in E_2$ **do**
    $K \leftarrow K \cup$ Kante–Kante–Kontakt$(e_1, e_2, \mathbf{d}_1, \mathbf{d}_2)$

Die Laufzeitanalyse ergibt für zwei Polyeder P_1 und P_2

$$\begin{aligned} & \sum_{v_1 \in V_1} \sum_{f_2 \in F_2} O(|f_2|) + \sum_{v_2 \in V_2} \sum_{f_1 \in F_1} O(|f_1|) + \sum_{e_1 \in E_1} \sum_{e_2 \in E_2} O(1) \\ = & \sum_{v_1 \in V_1} O(|P_2|) + \sum_{v_2 \in V_2} O(|P_1|) + \sum_{e_1 \in E_1} O(|P_2|) \\ = & O(|P_1| \cdot |P_2|) + O(|P_1| \cdot |P_2|) + O(|P_1| \cdot |P_2|) \\ = & O(|P_1| \cdot |P_2|) \end{aligned}$$

Damit läßt sich die Laufzeitkomplexität der Berechnung der möglichen Kontaktsituationen zwischen zwei Polyedern zusammenfassen in folgendem Lemma:

Lemma 3.4 :

> *Für zwei Polyeder P_1 und P_2, die nicht notwendigerweise konvex sein müssen, sowie eine Translationsebene e, parallel zu der sich P_1 bewegt, kann in Zeit $O(|P_1| \cdot |P_2|)$ die Menge aller möglichen Kontaktsituationen zwischen den beiden Polyedern parametrisiert bzgl. der Richtungsvektoren der Translationsebene e berechnet werden. Dabei ergeben sich $O(|P_1| \cdot |P_2|)$ Kontaktsegmente.*

3.1.2 Berechnung der Konfigurationsraumhindernisse

Im vorhergehenden Abschnitt wurde gezeigt, wie man aus der Beschreibung der Polyeder die Menge der Liniensegmente berechnet, die die Konfigurationsraumhindernisse beschreiben. Dabei wurden alle lokalen Kontaktmöglichkeiten zwischen Polyedern betrachtet, so daß i.a. auch Liniensegmente berechnet wurden, die zur Beschreibung der Begrenzung der Konfigurationsraumhindernisse nicht beitragen. Darüberhinaus wird bei den Polyedern nicht vorausgesetzt, daß diese konvex sind, so daß einzelne Kontakte, die die Begrenzung der Konfigurationsraumhindernisse beschreiben, ineinander übergehen können, d.h. daß sich die zugehörigen Liniensegmente im Konfigurationsraum schneiden.

In diesem Abschnitt wird nun gezeigt, wie man aus der Menge der sich gegenseitig schneidenden Kontaktsegmente die Beschreibung der Konfigurationsraumhindernisse berechnet. Konfigurationsraumhindernisse in der Ebene sind Polygone bzw. polygonale Regionen mit geschlossenem Rand, zu deren Beschreibung die Liste ihrer Eckpunkte im Gegenuhrzeigersinn berechnet werden muß. Es stellt sich somit folgendes Problem:

gegeben:

K	Liste von Liniensegmenten im $\mathbb{R}^2$
$\mathbf{p}_{start}$	Startposition der Bewegung des bewegten Polyeders

gesucht:

L_{CO}	Liste der Konfigurationsraumhindernisse, die die Zusammenhangskomponente des Freiraum beschreibt, die $\mathbf{p}_{start}$ enthält

An dieser Stelle wird benötigt, daß die Startposition des bewegten Polyeders frei ist. Dadurch ist sichergestellt, daß $\mathbf{p}_{start}$ im Konfigurationsraum im Freiraum liegt, und damit das Innere der Konfigurationsraumhindernisse definiert ist.

Bemerkung 3.5 :

An dieser Stelle wird sichtbar, daß zur Lösung des Bewegungsplanungsproblems nicht die Beschreibung des gesamten Freiraums des Konfigurationsraumes benötigt wird, sondern nur die Beschreibung der Zusammenhangskomponente des Freiraums, die die Startposition der Bewegung enthält, da alle anderen Punkte des Konfigurationsraumes von der Startposition aus nicht erreichbar sind. Liegt die Endposition der Bewegung in dieser Zusammenhangskomponente, ist das Bewegungsplanungsproblem lösbar, anderenfalls hat es keine Lösung.

Dieses Problem wurde von L. Guibas, M. Sharir und S. Sifrony in [GSS89] behandelt. Dabei wurde zum einen die Komplexität der Beschreibung der Zusammenhangskomponente abgeschätzt und zum anderen ein Algorithmus zur Berechnung der Zusammenhangskomponente angegeben. Dabei wird gezeigt, das die Beschreibung der Zusammenhangskomponente eine Davenport-Schinzel-Sequenz ist. Daher läßt sich deren Komplexität abschätzen durch die Länge $\lambda_s(n)$ der längsten (n,s)-Davenport-Schinzel-Sequenz, wobei n die Anzahl der Bewegungsbeschränkungen, d.h. die Anzahl der Kontaktsegmente, und s ein Parameter abhängig von der Art der Bewegungsbeschränkungen ist.

$$s = \text{Max. Anzahl Schnitte zwischen zwei Bewegungsbeschränkungen} + 2$$

Für den Fall von Kontaktsegmenten, d.h. Liniensegmenten als Bewegungsbeschränkungen, ergibt sich eine Komplexität der Beschreibung der Zusammenhangskomponente von $O(\lambda_3(n)) = O(n \cdot \alpha(n))$, wobei $\alpha(n)$ die inverse Ackermannfunktion ist, die bekanntermaßen extrem langsam wächst, so daß die Komplexität einer Zusammenhangskomponente „quasilinear"ist in der Anzahl der Kontaktsegmente.

3.1.2.1 Berechnung der Zusammenhangskomponente

Die kanonische Vorgehensweise besteht aus zwei Teilen:

1. Zerlegung der Kontaktsegmente in Teile, die sich nur in ihren Endpunkten schneiden
2. Berechnung der Zusammenhangskomponente aus den entstandenen Liniensegmenten

Zerlegung der Kontaktsegmente

Zur Zerlegung von Liniensegmenten in sich nur in den Endpunkten schneidende Teile existieren Standard-Algorithmen wie der Plane-Sweep-Algorithmus von J. Bentley und T. Ottmann [BO79], der sich durch eine einfache Beschreibung auszeichnet und dadurch für eine praktische Implementierung geeignet ist.

Die Laufzeit des Algorithmus ist output-sensitiv mit $O((n + k) \cdot \log n)$, wobei n die Anzahl der Liniensegmente in der Eingabe und k die Anzahl der Schnittpunkte zwischen den Liniensegmenten ist. Im worst-case gibt es zwischen n Liniensegmenten $O(n^2)$ Schnittpunkte, so daß die worst-case-Laufzeit des Algorithmus $O(n^2 \cdot \log n)$ beträgt.

Berechnung der Zusammenhangskomponente aus den zerlegten Segmenten

Als Eingabe für den Algorithmus liegt nun eine Menge von Liniensegmenten vor, die sich gegenseitig nicht schneiden außer in ihren Endpunkten und deren äußere Begrenzung eine Menge von Polygonen und evtl. einer polygonalen Region beschreibt. Aus diesen soll das Aussehen der Zusammmenhangskomponente berechnet werden, die die Startposition der Bewegung enthält.

Dazu könnten, wie im Fall der Zerlegung der Kontaktsegmente, zwei Standard-Algorithmen verwendet werden, ein Plane-Sweep-Algorithmus zur Berechnung aller Zusammenhangskomponenten der Zerlegung der Ebene durch die Liniensegmente und ein Point-Location Algorithmus zur Berechnung der Zusammenhangskomponente, in der sich die Startposition der Bewegung befindet. Dies wäre jedoch zu aufwendig, so daß an dieser Stelle ein Algorithmus verwendet wird, der, ausgehend von der Startposition der Bewegung, das Aussehen nur dieser Zusammenhangskomponente berechnet.

Dazu wird ein Rotations-Sweep-Algorithmus mit Rotationszentrum in der Startposition der Bewegung verwendet. Als Sweep-Line wird dabei ein Strahl ausgehend von der Startposition der Bewegung benutzt.

Um die topologische Information der Liniensegmente zu fassen, wird aus der Menge von Liniensegmenten ein Graph aufgebaut, dessen Knoten den Endpunkten der Liniensegmente und dessen Kanten die Liniensegmente selbst entsprechen. Werden dabei die erzeugten Knoten in einem balancierten Suchbaum abgespeichert, kann der Aufbau des Graphen für n Liniensegmente in Zeit $O(n \cdot \log n)$ erfolgen.

Der Algorithmus rotiert nun einmal um die gesamte Szene und berechnet dabei aus dem Graphen die äußere Begrenzung der Konfigurationsraumhindernisse, indem er für jedes Hindernis ausgehend von einer orientierten Kante das Hindernis umrundet.

Berechnung eines Konfigurationsraumhindernisses

Ausgehend von einer gerichteten Kante, die sicher zur äußeren Begrenzung des Konfigurationsraumhindernisses gehört, wird die gesamte äußere Begrenzung des Konfigurationsraumhindernisses berechnet, indem man aus den Nachfolgekanten sukzessive diejenige Kante auswählt, die mit der Vorgängerkante den größten Innenwinkel einschließt.

Die Entscheidung, welche Nachfolgekante mit der gegebenen den größten Innenwinkel einschließt, kann ohne Benutzung der trigonometrischen Funktionen ermittelt werden durch zwei Tests, die für jede Nachfolgekante durchgeführt werden. Sei $\mathbf{e}_r$ der durch die Kante e definierte normierte Richtungsvektor in der Ebene, dann gilt:

$$\varphi(\mathbf{e}_r, \mathbf{e}_r^{'}) > 180° \Leftrightarrow (\mathbf{e}_r^{\perp})^T \mathbf{e}_r^{'} < 0$$

wobei

$$\mathbf{e}^{\perp} = \begin{bmatrix} e_2 \\ -e_1 \end{bmatrix} \text{ für } \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

Mit der Gleichung

$$(-\mathbf{e}_r)^T \mathbf{e}_r^{'} = |\mathbf{e}_r||\mathbf{e}_r^{'}| \cos\varphi(-\mathbf{e}_r, \mathbf{e}_r^{'}) \overset{|\mathbf{e}_r|=|\mathbf{e}_r^{'}|=1}{=} \cos\varphi(-\mathbf{e}_r, \mathbf{e}_r^{'})$$

und der Tatsache, daß die Cosinus-Funktion im Intervall $[0°, 180°]$ streng monoton fallend und im Intervall $[180°, 360°]$ streng monoton wachsend ist, läßt sich die Kante mit maximalem Innenwinkel berechnen durch

$$e_{max} = \begin{cases} \arg\left(\max\limits_{e' \in \operatorname{succ}(e) \text{ mit } (\mathbf{e}_r^{\perp})^T \mathbf{e}_r^{'} < 0} -\mathbf{e}_r^T \mathbf{e}_r^{'} \right) & \text{falls} \quad \exists\, e' \in \operatorname{succ}(e) \text{ mit } (\mathbf{e}_r^{\perp})^T \mathbf{e}_r^{'} < 0 \\ \arg\left(\min\limits_{e' \in \operatorname{succ}(e)} -\mathbf{e}_r^T \mathbf{e}_r^{'} \right) & \text{sonst} \end{cases}$$

Die Berechnung ist in der Abbildung 3.3 graphisch veranschaulicht.

Mit Hilfe dieser Berechnung erhält man eine Folge von gerichteten Kanten, die die Begrenzung des Konfigurationsraumhindernisses beschreiben. Alle Kanten, die an Knoten des Graphen beginnen oder enden, die zu dem berechneten Konfigurationsraumhindernis gehören, selbst aber nicht zu der Beschreibung beitragen, werden aus dem Graphen gestrichen.

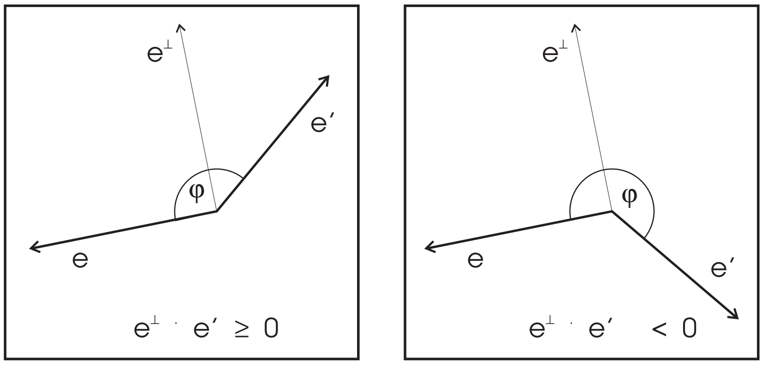

Abbildung 3.3: Berechnung der Nachfolgekante in der Hindernisbegrenzung

Bei der Berechnung des Konfigurationsraumhindernisses wird jedes Liniensegment einer Menge von zusammenhängenden Liniensegmenten, die das Hindernis beschreiben, nur konstant oft betrachtet. Die Berechnung des Innenwinkels und das Streichen aus dem Graphen kann in konstanter Zeit erfolgen, so daß die Gesamtlaufzeit zur Berechnung eines Konfigurationsraumhindernisses, das von n zusammenhängenden Liniensegmenten beschrieben wird, ausgehend von einer gerichteten Kante in Zeit $O(n)$ erfolgen kann.

Der gesamte Rotations-Sweep-Algorithmus benutzt nun diese Berechnung der Konfigurationsraumhindernisse ausgehend von einer gerichteten Kante als Funktion in einer Transition.

Als Basis-Sweep-Strahl dient ein horizontaler Strahl ausgehend von der Startposition der Bewegung im Konfigurationsraum in negativer x-Richtung; die Rotation des Sweep-Strahls erfolgt im Uhrzeigersinn. Durch diese Festlegung werden alle Liniensegmente gerichtet bzgl. des Sweep-Strahls. Startpunkt eines Liniensegmentes ist der Endpunkt, der während des Sweeps als erster Endpunkt erfaßt wird.

Während des Sweeps ist nun noch zu beachten, daß Konfigurationsraumhindernisse in Konfigurationsraumhindernissen liegen können und damit nicht zur Beschreibung der zu berechnenden Zusammenhangskomponente beitragen. Um diese nicht zu berechnen, wird in der y-Struktur die aktuelle Einteilung des Sweep-Strahles in freie und unfreie Intervalle mitgeführt.

Der Rotations-Sweep-Algorithmus ist nun wie folgt aufgebaut:

x-Struktur

Alle Start- und Endpunkte der Liniensegmente sortiert nach ihrem Winkel mit dem Basis-Sweep-Strahl

y-Struktur

- Menge aller Liniensegmente auf dem Sweep-Strahl, die zur Beschreibung von Konfigurationsraumhindernissen beitragen, sortiert nach ihrem Abstand vom Sweep-Zentrum

- Einteilung des Sweep-Strahles in Freiraum- und Hindernisintervalle

Transitionen

- Punkt ist Eckpunkt eines bereits bearbeiteten Hindernisses:
    1. Löschen der endenden Begrenzungskanten
    2. Einfügen evtl. beginnender Begrenzungskanten
    3. Falls in dem Eckpunkt zwei Begrenzungskanten enden, vereinige die drei Intervalle (Freiraum – Hindernisinneres – Freiraum) zu einem Freiraumintervall
- Punkt ist kein Eckpunkt eines bereits bearbeiteten Hindernisses:
    1. Berechne aus den in diesem Punkt startenden Liniensegmenten dasjenige, das mit dem Sweep-Strahl den größten Winkel im Uhrzeigersinn einschließt.
       Sei $\mathbf{v}$ der normierte Richtungsvektor des aktuellen Sweep-Strahls und $\mathbf{p}$ der aktuelle Transitionspunkt, so ergibt sich e_{Anfang} aus

       $$e_{Anfang} = \arg\left(\min_{e \text{ startet in } \mathbf{p}} \mathbf{v}^T \mathbf{e}_r\right)$$
    2. Entscheide, ob das Konfigurationsraumhindernis in einem bereits berechneten liegt Dies ist möglich, da die aktuell in dem Sweep-Strahl befindlichen Liniensegmente den Sweep-Strahl in alternierende Intervalle bestehend aus Freiraum und Hindernis unterteilen.
       Falls ja,
        (a) Lösche in dem Teilgraphen zu $\mathbf{p}$ alle zugehörigen Transitionspunkte
        (b) Lösche die Zusammenhangskomponente des Graphen, die $\mathbf{p}$ enthält

       Falls nein,
        (a) Berechne mit Hilfe der gerichteten Kante das Konfigurationsraumhindernis
        (b) Lösche alle zu dem Teilgraphen gehörigen Transitionspunkte, die Knoten entsprechen, die nicht zur Beschreibung des Konfigurationsraumhindernisses beitragen.
        (c) Füge die beiden Kanten, die in $\mathbf{p}$ starten, in den Sweep-Strahl ein
        (d) Markiere alle Eckpunkte der Hindernisbegrenzung als bearbeitet
        (e) Teile das Freiraumintervall durch die beiden Begrenzungskanten in drei Intervalle: Freiraum – Hindernisinneres – Freiraum

Zur Berechnung der initialen y-Struktur wird der Schnitt aller Liniensegmente mit dem initialen Sweep-Strahl berechnet. Für alle Liniensegmente, die den Sweep-Strahl schneiden, wird nach aufsteigender Sortierung ihres Abstandes zum Sweep-Zentrum sukzessive folgende Berechnung durchgeführt:

1. Liegt das Liniensegment im Inneren eines bereits berechneten Hindernisses?
   Falls ja,

   Streiche die zu dem Liniensegment gehörige Zusammenhangskomponente des Graphen und die zugehörigen Transitionspunkte aus der x-Struktur

Falls nein,

(1) Berechne das zu dem Liniensegment gehörige Konfigurationsraumhindernis
(2) Füge alle zu Kanten des Konfigurationsraumhindernisses gehörenden Liniensegmente, die den aktuellen Sweep-Strahl schneiden, in die y-Struktur ein
(3) Unterteile das bestehende einseitig unbeschränkte Freirauminterval in Hindernis- und Freiraumintervalle
(4) Streiche alle Transitionspunkte der Zusammenhangskomponente des Graphen aus der x-Struktur, die nicht zur Beschreibung des Konfigurationsraumhindernisses beitragen
(5) Streiche aus den Transitionspunkten, die Ecken des Konfigurationsraumhindernisses entsprechen, die Kanten, die nicht zur Beschreibung der Hindernisbegrenzung beitragen

Die Vorgehensweise des Algorithmus ist beispielhaft in Abbildung 3.4 graphisch veranschaulicht.

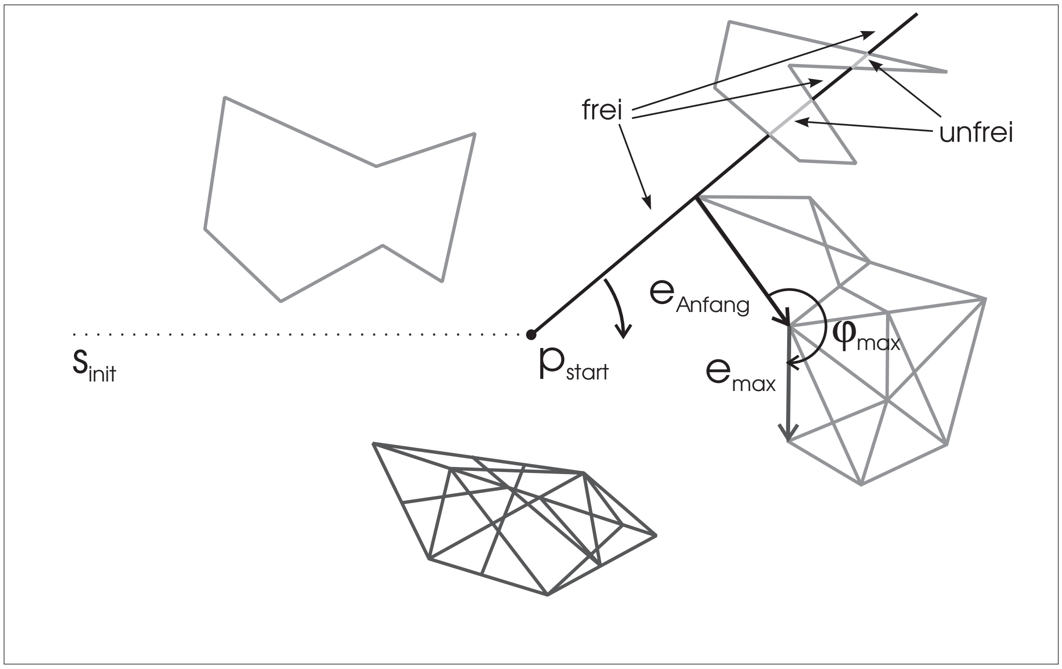

Abbildung 3.4: Rotations-Sweep zur Berechnung einer Zusammenhangskomponente des Freiraums

Laufzeitanalyse

Die Berechnung des Graphen mit der topologischen Information der Kanten kann mit Hilfe eines balancierten Suchbaumes für die zu den Endpunkten der Liniensegmente gehörenden Knoten des Graphen für n Liniensegmente in Zeit $O(n \cdot \log n)$ erfolgen.

Bei der Berechnung eines Konfigurationsraumhindernisses wird genau eine Zusammenhangskomponente des Graphen betrachtet und jede Zusammenhangskomponente des Graphen wird im Verlauf des Rotations-Sweeps genau einmal betrachtet. Die Laufzeit zur Berechnung eines Konfigurationsraumhindernisses aus einer Zusammenhangskomponente ist linear in der Größe der Zusammenhangskomponente, so daß sich eine Laufzeit zur Berechnung aller Konfigurationsraumhindernisse ausgehend von einer gerichteten Kante von $O(n)$ ergibt, wenn n Liniensegmente als Eingabe vorliegen.

Für den Rotations-Sweep gilt, daß es bei n Liniensegmenten höchstens $O(n)$ Transitionspunkte geben kann und daß jedes Liniensegment höchstens einmal in die y-Struktur eingefügt oder entfernt wird bzw. die zugehörigen Transitionspunkte aus der x-Struktur gestrichen werden. Während des Sweeps kann die Modifikation der Intervallstruktur auf dem Sweep-Strahl in Zeit O(1) erfolgen. Insgesamt ergibt sich somit eine Gesamtlaufzeit für die Durchführung des Rotations-Sweeps ohne Berechnung der Konfigurationsraumhindernisse von $O(n \cdot \log n)$.

Die Berechnung der initialen y-Struktur ohne die Berechnung der Konfigurationsraumhindernisse kann in Zeit $O(n \cdot \log n)$ erfolgen, da die Berechnung der Schnittpunkte aller Liniensegmente mit dem Sweep-Strahl und die Modifikation der Intervalleinteilung in Linearzeit und das Einfügen der Liniensegmente in die aktuelle y-Struktur in Zeit $O(n \cdot \log n)$ erfolgen kann.

Insgesamt ergibt sich bei einer Eingabe von n Liniensegmenten eine Gesamtlaufzeit von:

$$
\begin{aligned}
& \underbrace{O(n \cdot \log n)}_{\text{Berechnung des Graphen}} + \underbrace{O(n \cdot \log n)}_{\text{Initialisierung der x-Struktur}} + \\
& \underbrace{O(n \cdot \log n)}_{\text{Initialisierung der y-Struktur}} + \underbrace{O(n \cdot \log n)}_{\text{Durchführung Rotations-Sweep}} + \\
& \underbrace{O(n)}_{\text{Berechnung der Konfigurationsraumhindernisse}} \\
= \; & O(n \cdot \log n)
\end{aligned}
$$

Die spezielle Problematik bei der Berechnung der Zusammenhangskomponenten liegt darin, daß nur die Schnittpunkte zwischen Liniensegmenten berechnet werden sollen, die zur Beschreibung der Zusammenhangskomponente notwendig sind.

Dazu wird in [GSS89] eine divide-and-conquer Vorgehensweise vorgeschlagen. Dabei werden in jedem Schritt zwei bereits berechnete Zusammenhangskomponenten zu einer zusammengemischt, d.h. aus dem Schnitt von zwei Zusammenhangskomponenten diejenige berechnet, die die Startposition der Bewegung enthält.

In jedem Mischschritt zweier Zusammenhangskomponenten wird die neue Zusammenhangskomponente durch zwei Plane-Sweep-Algorithmen, einen in positiver x-Richtung und einen in negativer x-Richtung, berechnet. Bei jedem dieser beiden Plane-Sweep-Algorithmen beginnt die Berechnung der Zusammenhangskomponente erst nach dem Erreichen der Startposition. Vor dem Erreichen der Startposition werden die beiden berechneten Zusammenhangskomponenten in getrennten y-Strukturen verwaltet und somit eine Berechnung unnötiger Schnittpunkte zwischen den beiden Zusammenhangskomponenten vermieden.

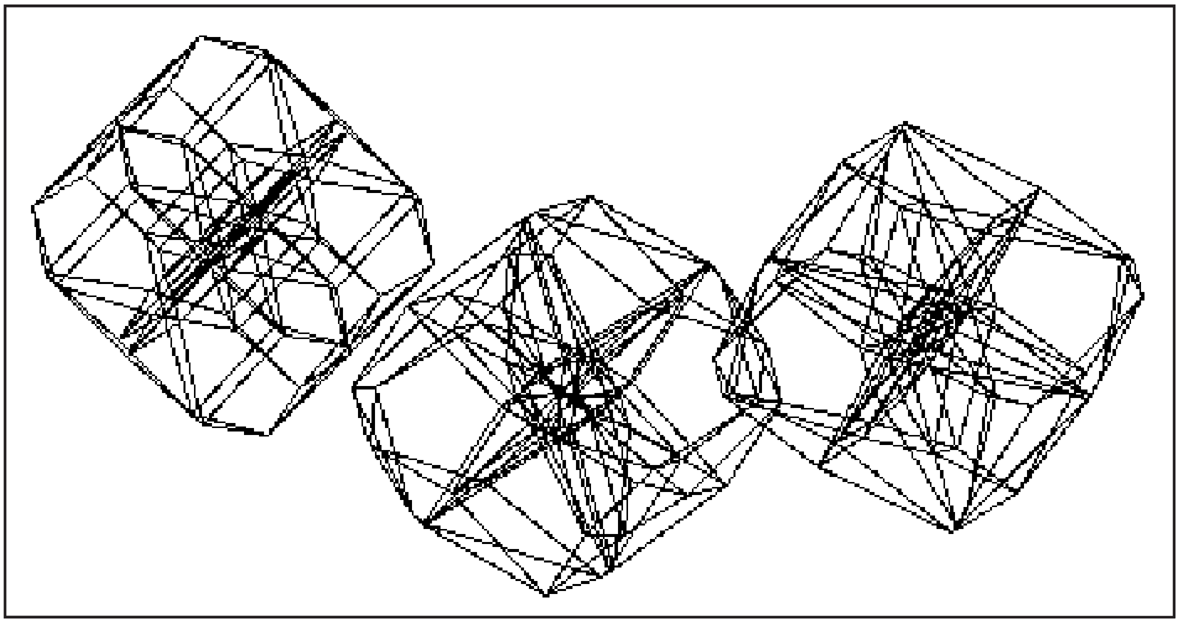

Abbildung 3.5: Kontaktsegmente im Konfigurationsraum

Nach dem Erreichen der Startposition können x-monotone Teile der Zusammenhangskomponente berechnet werden, wobei nur die zur Beschreibung der neuen Zusammenhangskomponente benötigten Schnittpunkte zwischen den beiden bereits berechneten Zusammenhangskomponenten berechnet werden. Dieser Vorteil wird durch einen relativ großen Überwachungsaufwand von Liniensegmenten bezahlt, der sich nicht in der theoretischen Gesamtlaufzeit niederschlägt, aber die Beschreibungskomplexität einer Transition des Plane-Sweep-Algorithmus erhöht.

Die einzelnen x-monotonen Teile der Zusammenhangskomponenten können während des Plane-Sweep-Algorithmus in einem Graphen verwaltet werden, der die Nachbarschaftsbeziehungen der x-monotonen Teile der Zusammenhangskomponente ausdrückt. Da das Aussehen der Zusammenhangskomponente in der Startposition bekannt ist, kann der durch den ersten Plane-Sweep aufgebaute Graph durch den zweiten Plane-Sweep in die umgekehrte Richtung erweitert werden, so daß man am Ende der beiden Plane-Sweep-Algorithmen einen Graphen von x-monotonen Teilen der neuen Zusammenhangskomponente erhält, aus dem mit Hilfe der Nachbarschaftsbeziehung das gesamte Aussehen der Zusammenhangskomponente berechnet werden kann. Dadurch kann die Beschreibung einer Zusammenhangskomponente, die die Startposition der Bewegung enthält, aus einer Menge von Liniensegmente in Zeit $O(n \cdot \alpha(n) \cdot \log^2(n))$ berechnet werden.

Für den Fall von Liniensegmenten als Bewegungsbeschränkungen gibt es auch einen randomisierten Algorithmus von B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, J. Snoeyink [CEG+93], der in erwarteter Laufzeit von $O(n \cdot \alpha(n) \cdot \log n)$ diese Zusammenhangskomponente berechnet.

Im Hinblick auf die praktische Implementierung bereitet der Algorithmus in [GSS89] wegen der Beschreibungskomplexität der einzelnen Transitionen und des Berechnungsaufwandes der Beschreibung der Zusammenhangskomponente aus verschiedenen x-monotonen Teilen einige Probleme. Daher wurde eine alternative Strategie implementiert, mit der man die gesuchte Zusammenhangskomponente berechnen kann, da in diesem Teil der Implementation relativ starke numerische Probleme aufgetreten sind, die zu beherrschen mit Hilfe des komplexen Algorithmus von [GSS89] sehr schwierig gewesen wäre.

Der Nachteil dieser Vorgehensweise gegenüber der in [GSS89] beschriebenen ist, daß zunächst alle Schnittpunkte zwischen den Liniensegmenten berechnet werden müssen. Das kann dazu führen, daß viele Schnittpunkt im Inneren der Konfigurationsraumhindernisse berechnet werden, wie in Abbildung 3.5 zu sehen, die nicht zur Beschreibung der Konfigurationsraumhindernisse benötigt werden.

Um die Berechnung möglichst vieler Schnittpunkte von Liniensegmenten im Inneren von Konfigurationsraumhindernissen zu vermeiden, ist eine Strukturierung für die Kontaktsegmente gesucht, die innere Liniensegmente möglichst frühzeitig aus der Betrachtung streicht, und damit die Anzahl der berechneten inneren Schnittpunkte reduziert.

3.1.2.2 Strukturierung der Kontaktsegmente

Zur Strukturierung der Kontaktsegmente muß erneut die Berechnung der Kontaktsegmente betrachtet werden.

Im Abschnitt **Berechnung der lokalen Kontaktpositionen** wurden die Grundlagen zur Berechnung der Kontaktsegmente gelegt. Dabei wurde davon ausgegangen, daß das bewegte Polyeder und das feste Polyeder immer als Ganzes betrachtet wird. Man kann jedoch auch nur Teile der Polyeder betrachten und für diese die Kontaktsegmente berechnen.

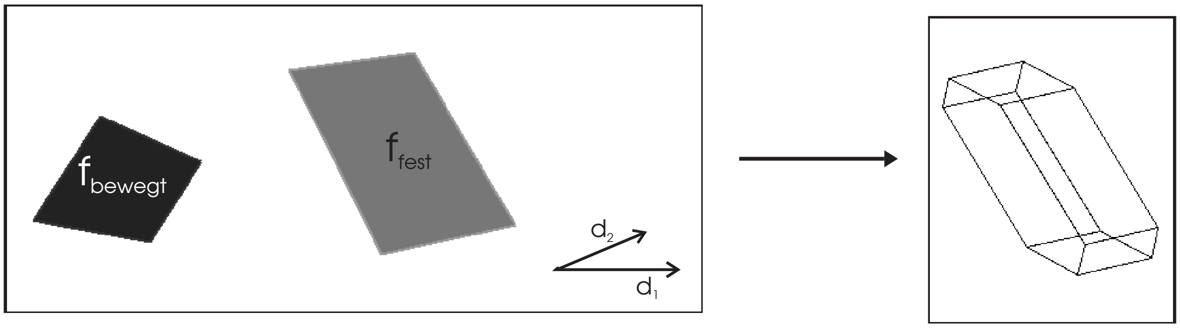

Abbildung 3.6: Konfigurationsraumhindernis bei Betrachtung zweier Flächen

Dabei sieht man, daß die Berechnung der Kontaktsegmente zwischen zwei Flächen zu einer Beschreibung von Polygonen führt, wie in Abbildung 3.6 zu sehen.

Betrachtet man die aus diesen Vergleichen hervorgehenden Kontaktsegmente und die resultierenden Konfigurationsraumhindernisse als Basiskonfigurationsräume, ergibt sich das Aussehen des Freiraumes des gesamten Konfigurationsraumes aus dem Schnitt der Freiräume der Basiskonfigurationsräume.

Daher bietet sich eine inkrementelle Vorgehensweise zur Berechnung der Zusammenhangskomponente des gesamten Konfigurationsraumes an, bei der sukzessive die einzelnen Zusammenhangskomponenten baumartig zusammengemischt werden, so daß sich eine binäre Strukturierungshierarchie für die Zusammenhangskomponente des gesamten Konfigurationsraumes ergibt, wie in Abbildung 3.7 zu sehen.

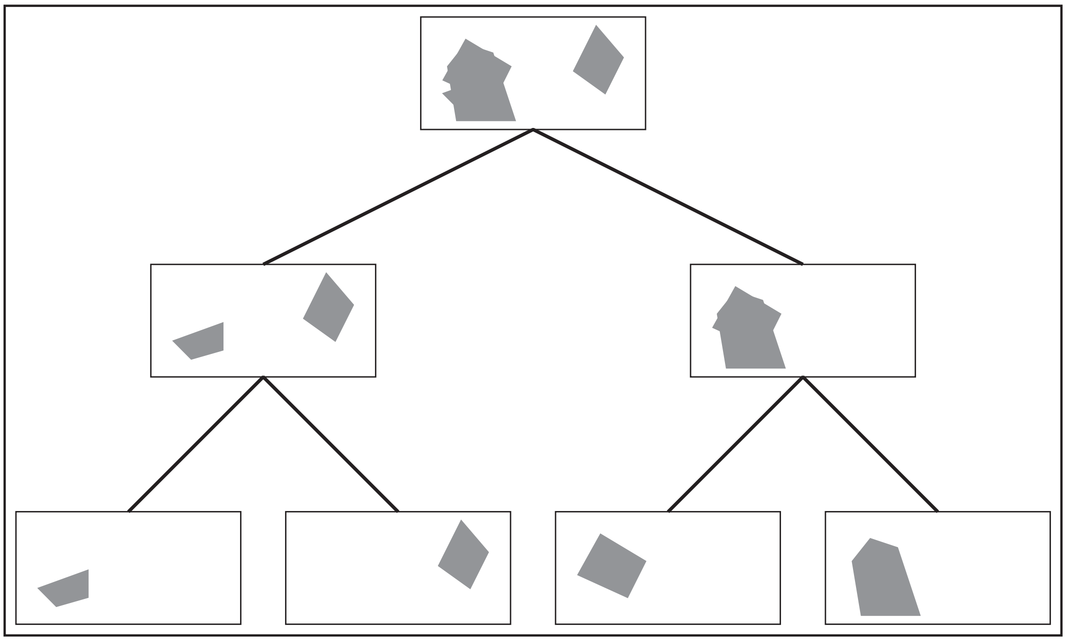

Abbildung 3.7: Strukturierungshierarchie für die Konfigurationsraumhindernisse

Durch die Strukturierung werden innere Liniensegmente, die zur Beschreibung der Konfigurationsraumhindernisse nicht beitragen, möglichst frühzeitig aus der Betrachtung gestrichen, so daß eine geringere Anzahl von inneren Schnittpunkten zwischen Liniensegmenten berechnet wird.

Folgende Einteilungen von Objekten bieten sich dabei zur Strukturierung der Kontaktsegmente in Basiskonfigurationsräume an:

1. bewegtes Objekt gegen ein Hindernis
2. ein Fläche des bewegten Objektes gegen ein Hindernis
3. ein Fläche des bewegten Objektes gegen eine Fläche des Hindernisses

Testet man nun Teile von Objekten gegeneinander, um die Kontaktsegmente zu berechnen, die Konfigurationsraumhindernisse beschreiben, werden Kontaktsegmente mehrfach erzeugt, da Kante–Kante–Kontakte und Ecke–Fläche–Kontakte mehrfach betrachtet werden.

Diese werden jedoch während der Berechnung der Zusammenhangskomponente des gesamten Konfigurationsraums wieder zusammengeführt.

Für die Anzahl der bei der Berechnung der Basiskonfigurationsräume berechneten Kontaktsegmente gilt:

1. bewegtes Objekt gegen Hindernis:
   $|F_{bewegt}| \cdot |V_{fest}| + |V_{bewegt}| \cdot |F_{fest}| + |E_{bewegt}| \cdot |E_{fest}| = O(|P_{bewegt}| \cdot |P_{fest}|)$
2. bewegte Fläche gegen Hindernis:
   $\sum_{i=1}^{|F_{bewegt}|} (|f_i| \cdot (|F_{fest}| + |V_{fest}| + |E_{fest}|)) = O(|P_{bewegt}| \cdot |P_{fest}|)$
3. bewegte Fläche gegen feste Fläche:
   $\sum_{i=1}^{|F_{bewegt}|} \sum_{j=1}^{|F_{fest}|} \left(|V_{f_i}| + |V_{f_j}| + |E_{f_i}| \cdot |E_{f_j}|\right) = \sum_{i=1}^{|F_{bewegt}|} \sum_{j=1}^{|F_{fest}|} (|f_i| + |f_j| + |f_i| \cdot |f_j|)$
   $= O(|P_{bewegt}| \cdot |P_{fest}|)$

D.h. die Anzahl der insgesamt berechneten Kontaktsegmente vergrößert sich durch die Strukturierung der Polyeder bei der Berechnung von Kontaktsegmenten nicht.

Es hat sich jedoch gezeigt, daß in praktischen Beispielen vielfach sich gegenseitig überlappende und identische Liniensegmente auftreten. Werden alle Kontaktsegmente in einem Schritt betrachtet, werden überlappende Liniensegmente vorweg in sich nicht schneidende Teile unterteilt und mehrfach vorhandene Liniensegmente eliminiert. In der hierarchischen Vorgehensweise befinden sich i.a. solche überlappenden oder identischen Liniensegmente in verschiedenen Teilbäumen und treffen erst im Laufe des Mischprozesses aufeinander, d.h. es werden dadurch Schnittpunkte von Liniensegmenten mehrfach berechnet, die bei einer direkten Betrachtung aller Kontaktsegmente nur einmal berechnet werden.

Dadurch konnte keine Strukturierung der Kontaktsegmente gefunden werden, die für alle Beispiele vorteilhaft gewesen wäre. Vielmehr hängt die Laufzeitveränderung durch die Strukturierung vom konkreten Beispiel ab.

Laufzeitbetrachtung für die Berechnung der Konfigurationsraumhindernisse

Sind n Kontaktsegmente gegeben, kann die Menge der Liniensegmente, die sich nur noch in den Endpunkten schneiden, in Zeit $O(n^2 \cdot \log n)$ berechnet werden. Dabei entstehen höchstens $O(n^2)$ zerlegte Liniensegmente. Aus diesen $O(n^2)$ Liniensegmenten kann in Zeit $O(n^2 \cdot \log n^2) = O(n^2 \cdot \log n)$ die Zusammenhangskomponente berechnet werden, die die Startposition der Bewegung enthält. Insgesamt gilt:

Lemma 3.6:

> *Gegeben seien n Liniensegmente in der Ebene, die die Begrenzung der Konfigurationsraumhindernisse eines Bewegungsplanungsproblems mit zwei Freiheitsgraden beschreiben und ein beliebiger Punkt $\mathbf{x}$ im Konfigurationsraum, der nicht auf einem der Liniensegmente liegt. Dann hat die Zusammenhangskomponente, die $\mathbf{x}$ enthält, höchstens Komplexität $O(n \cdot \alpha(n))$ und kann in Zeit $O(n^2 \cdot \log n)$ berechnet werden.*

3.2 Berechnungen im Konfigurationsraum

Aus der **Berechnung des Konfigurationsraumes** erhält man eine Beschreibung der Zusammenhangskomponente des Freiraums des Konfigurationsraumes, die den Startpunkt der Bewegung enthält, in Form einer Liste von Konfigurationsraumhindernissen. Dies sind einfache Polygone und evtl. eine polygonale Region, die die Zusammenhangskomponente nach außen hin beschränkt.

Um nun eine Lösung für das Bewegungsplanungsproblem zu erhalten, muß in diesem Konfigurationsraum ein Weg von der Start- zur Endposition der Bewegung berechnet werden, die das Innere der Konfigurationsraumhindernisse vermeidet.

Bemerkung 3.7 :

> *Da die bewegten Objekte als abgeschlossene Mengen und die festen Objekte als offene Mengen definiert sind, ergeben sich keine Kollisionen zwischen den Objekten, wenn das bewegte Objekt ein festes Objekt berührt.*
> *Diese Positionen des bewegten Objektes entsprechen den Begrenzungen der Konfigurationsraumhindernisse, so daß der zu suchende Weg im Konfigurationsraum die Begrenzung der Konfigurationsraumhindernisse benutzen darf.*

Der berechnete Konfigurationsraum erlaubt es an dieser Stelle zu entscheiden, ob eine Bewegung von der Startposition zu der Endposition möglich ist. Nach Voraussetzung ist die Startposition der Bewegung frei. Liegt die Endposition der Bewegung im Inneren eines der berechneten Konfigurationsraumhindernisse, ist eine Bewegung unmöglich, da die Endposition der Bewegung entweder nicht frei ist, oder, falls die Endposition frei ist, diese aufgrund von Hindernissen im Objektraum nicht erreichbar ist. Ist die Endposition der Bewegung nicht erreichbar, endet die Bewegungsplanung an dieser Stelle mit dem Ergebnis, daß eine Bewegung in diesem Objektraum bei den gegeben Freiheitsgraden für das bewegte Objekt nicht möglich ist.

Test: Punkt im Inneren eines Konfigurationsraumhindernisses

Es ergibt sich folgende Problemstellung:

gegeben:

$\mathbf{p}$ Punkt im $\mathbb{R}^2$
P einfaches Polygon im $\mathbb{R}^2$

gesucht:

$\mathbf{p} \in \mathrm{int}(P)$?

Dazu testet man zunächst, ob $\mathbf{p}$ auf einer Begrenzungskante von P liegt. Ist dies der Fall, liegt $\mathbf{p}$ nicht im Inneren von P. Ist dies nicht der Fall, wählt man ausgehend von $\mathbf{p}$ einen Strahl in beliebiger Richtung und berechnet die Anzahl der Schnitte mit den Begrenzungskanten von

P. Bei einer ungeraden Anzahl von Schnitten liegt $\mathbf{p}$ im Inneren von P, bei einer geraden außerhalb von P.

Bemerkung 3.8 :

> *Mit Hilfe der Zusammenhangskomponente des Freiraums im Konfigurationsraum, die die Startposition der Bewegung enthält, kann entschieden werden, ob eine Bewegung von der Start- zu der Endposition möglich ist.*

Existiert eine Bewegung von der Startposition zur Endposition der Bewegung, ist es nun die Aufgabe der Berechnung im Konfigurationsraum, einen möglichen Weg explizit zu berechnen. Die gestellte Aufgabe läßt sich wie folgt formalisieren:

gegeben:

$\mathbf{p}_{start}$	Startposition der Bewegung
$\mathbf{p}_{end}$	Endposition der Bewegung
L_{CO}	Liste von Konfigurationsraumhindernissen, die die ZHK des Freiraums mit $\mathbf{p}_{start}$ beschreibt
P	Konfigurationsraum für das Polyeder

gesucht:

$\pi : [\,0,1] \to P$ stetig: $\pi(0) = \mathbf{p}_{start}, \pi(1) = \mathbf{p}_{end}$ und $\pi(t) \notin int(CO) \;\; \forall\, CO \in L_{CO}$

Gesucht ist ein Weg von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$ im Konfigurationsraum, der das Innere der Konfigurationsraumhindernisse vermeidet. Mit Hilfe der berechneten Zusammenhangskomponente des Freiraums ist es möglich, verschiedene Wegearten für das bewegte Objekt zu berechnen, darunter qualitative Wege wie z.B. kürzeste oder sicherste.

Im Rahmen dieser Arbeit wurden im $\mathbb{I}\mathbb{C}\mathbb{O}_{\mathbb{R}}$-System zwei Lösungsmöglichkeiten für das Wegeproblem implementiert: ein Verfahren, das in möglichst kurzer Laufzeit einen Weg berechnet, und ein Verfahren, das euklidisch kürzeste Wege berechnet.

3.2.1 Direkter Weg

Bei der Berechnung des direkten Weges betrachtet man eine direkte Verbindungslinie von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$. Man berechnet die Schnittpunkte dieser Verbindungslinie mit den Konfigurationsraumhindernissen und erhält dadurch folgenden Weg:

while ($\mathbf{p}_{end}$ nicht erreicht) **do**
    Laufe auf der Verbindungslinie bis zum nächsten Schnittpunkt mit
    einem Hindernis oder $\mathbf{p}_{end}$
    if (Schnittpunkt mit Hindernis) **then**
        Laufe entlang der Hindernisbegrenzung zum nächsten Schnittpunkt

Benutzt man zur Berechnung des Weges den Graphen, der bei der Berechnung der Zusammenhangskomponente des Freiraums im Konfigurationsraum berechnet wurde, berechnet die Schnittpunkte der Verbindungslinie mit den zu den Graphkanten gehörigen Liniensegmenten, fügt die freien Teile der Verbindungslinie als Kanten in den Graphen ein und teilt die Kanten an den Schnittpunkten mit der Verbindungslinie, wird jede Kante höchstens einmal geteilt und die Verbindungslinie zerfällt in höchstens soviele Teile wie der Graph Kanten hat. Die Größe des Graphen bleibt durch diese Operationen linear in der Größe der Zusammenhangskomponente des Freiraums und mit Hilfe von Depth-First-Search oder Breadth-First-Search kann in Linearzeit ein Weg von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$ gefunden werden. Insgesamt ergibt sich somit für die Berechnung des direkten Weges bei einer Größe von n für die Zusammenhangskomponente des Konfigurationsraumes eine Laufzeit von

$$\begin{aligned} & \underbrace{O(n)}_{\text{Berechnung der Schnittpunkte}} + \underbrace{O(n)}_{\text{Aufteilen der Kanten}} \\ & + \underbrace{O(n)}_{\text{Einfügen der Verbindungssegmente}} + \underbrace{O(n)}_{\text{Graphensuche}} \\ = \ & O(n) \end{aligned}$$

Da auch der Graph in linearer Zeit aus der Beschreibung des Freiraums aufgebaut werden kann, ist folgendes Lemma gezeigt.

Lemma 3.9 :

> *Gegeben seien eine Zusammenhangskomponente des Freiraums eines Konfigurationsraumes ZHK^{FP} der Größe n und zwei Punkte $\mathbf{p}_{start}, \mathbf{p}_{end} \in ZHK^{FP}$. Dann kann eine Bewegung in ZHK^{FP} von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$ in Zeit $O(n)$ berechnet werden.*

3.2.2 Euklidisch kürzester Weg

Wie in [La91] gezeigt wird, verlaufen euklidisch kürzeste Wege auf dem Sichtbarkeitsgraphen im Konfigurationsraum. Daher ist eine mögliche Vorgehensweise zur Berechnung kürzester Wege folgende:

1. Berechnung des Sichtbarkeitsgraphen G_{vis}
2. Bewertung der Kanten des Sichtbarkeitsgraphen
3. Berechnung eines kürzesten Weges in G_{vis} von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$

Theoretisch ist eine schnellere Berechnung des euklidisch kürzesten Weges von $\mathbf{p}_{start}$ nach $\mathbf{p}_{end}$ möglich, bei der die Berechnung des Sichtbarkeitsgraphen vermieden wird. Dazu wird von J. Hershberger und S. Suri in [HS93] ein $O(n \cdot \log^2 n)$ Algorithmus vorgestellt. Im Hinblick auf die praktische Implementierbarkeit wurde im Rahmen dieser Arbeit jedoch auf die Berechnung des kürzesten Weges mit Hilfe des Sichtbarkeitsgraphen zurückgegriffen.

3.2.2.1 Berechnung des Sichtbarkeitsgraphen

Der Sichtbarkeitsgraph ist wie folgt definiert:

Definition 3.10 (Sichtbarkeitsgraph)

Gegeben sei ein Konfigurationsraum P beschrieben durch eine Menge H von polygonalen Hindernissen CO in der Ebene und zwei Punkte $\mathbf{p}_{start}$ und $\mathbf{p}_{end}$ in P. Der **Sichtbarkeitsgraph** *ist ein ungerichteter Graph $G_{vis} = (V_{vis}, E_{vis})$ mit*

$$\begin{aligned} V_{vis} &= \{\mathbf{v} \in P \mid \mathbf{v} \in V_{CO} \text{ und } CO \in H\} \cup \{\mathbf{p}_{start}, \mathbf{p}_{end}\} \\ E_{vis} &= \left\{\{\mathbf{v}_1, \mathbf{v}_2\} \in V_{vis} \times V_{vis} \mid segment(\mathbf{v}_1, \mathbf{v}_2) \cap int\left(\bigcup_{CO \in H} CO\right) = \emptyset\right\} \end{aligned}$$

Der Sichtbarkeitsgraph verbindet also alle Ecken von Konfigurationsraumhindernissen bzw. Start- und Endposition der Bewegung, deren direkte Verbindung das Innere der Konfigurationsraumhindernisse nicht schneidet. Für einen Konfigurationsraum mit n Ecken hat der zugehörige Sichtbarkeitsgraph somit worst-case Größe $O(n^2)$.

Eine naive Vorgehensweise für die Berechnung des Sichtbarkeitsgraphen testet für alle $O(n^2)$ potentiellen Kanten des Sichtbarkeitsgraphen, ob sie das Innere eines Konfigurationsraumhindernisses schneiden. Dazu benötigt man Zeit $O(n)$, so daß sich eine Gesamtlaufzeit von $O(n^3)$ ergibt.

Für eine schnellere Berechnung wurden verschiedene effiziente Algorithmen vorgeschlagen wie z.B. der von E. Welzl [We85], der in worst-case-optimaler Zeit von $O(n^2)$ den Sichtbarkeitsgraph für n Liniensegmente berechnet, oder der von Gosh und Mount [GM87], dessen Laufzeit abhängig ist von der Größe des berechneten Sichtbarkeitsgraphen mit $O(k + n \cdot \log n)$, wobei k die Größe des berechneten Sichtbarkeitsgraphen ist.

Für die praktische Implementierung des Sichtbarkeitsgraphen im Rahmen dieser Arbeit innerhalb des IGOR-Systems wurde der Algorithmus von E. Welzl benutzt, da er neben einer einfachen Beschreibung des Algorithmus die Möglichkeit bietet, bei numerischen Problemen während der Ausführung des Algorithmus die Berechnung der verbleibenden Sichtbarkeitskanten aus einer Menge von potentiellen Sichtbarkeitskanten mit Hilfe der kanonischen Vorgehensweise fortzusetzen.

Die Idee des Algorithmus ist die, daß alle potentiellen Sichtbarkeitskanten berechnet und diese in einer geeigneten – einer Sortierung ähnlichen – Reihenfolge, die in Zeit $O(n^2)$ berechnet werden kann, daraufhin getestet werden, ob sie im Sichtbarkeitsgraphen enthalten sind. Dazu wird eine Sichtbarkeitsfunktion $vis: \bigcup_{CO \in H} V_{CO} \cup \{\mathbf{p}_{start}, \mathbf{p}_{end}\} \to \bigcup_{CO \in H} E_{CO}$ aufrechterhalten, mit Hilfe derer man in Zeit $O(1)$ entscheiden kann, ob eine Kante im Sichtbarkeitsgraph enthalten ist. Desweiteren wird diese Sichtbarkeitsfunktion in jedem Test einer potentiellen Sichtbarkeitskante in Zeit $O(1)$ modifiziert.

Die Funktionsweise des Algorithmus im einzelnen:

1. Berechne alle potentiellen Sichtbarkeitskanten (Richte die Kanten so, daß sie angesetzt im Nullpunkt des Koordinationsystems im ersten oder vierten Quardranten verlaufen)

2. Berechne eine gutartige Permutation der Kanten (In der Implementierung wurden die Kanten gemäß ihrer Steigung aufsteigend sortiert)

3. Berechne eine initiale Sichtbarkeitsfunktion vis, indem mit Hilfe eines Standard-Plane-Sweep-Algorithmus die Kante berechnet wird, die von der Ecke aus zu sehen ist, wenn der Blick vertikal nach unten, d.h. in negativer y-Richtung, gerichtet ist.

4. Durchlaufe die geordnete Folge von potentiellen Sichtbarkeitskanten, teste, ob sie im Sichtbarkeitsgraphen enthalten sind, und modifiziere die Sichtbarkeitsfunktion. Für die Modifikation der Sichtbarkeitsfunktion wird ausgenutzt, daß diese sich nur lokal für den Startpunkt einer Sichtbarkeitskante ändert.

Für den Test der $O(n^2)$ Sichtbarkeitskanten ergibt sich jeweils eine der folgenden Situationen (vgl. dazu Abbildung 3.8). Seien dabei $\mathbf{v}$ und $\mathbf{v}'$ die Endpunkte einer potentiellen Sichtbarkeitskante.

Fall 1: $\mathbf{v}$ und $\mathbf{v}'$ sind Endpunkte desselben Liniensegmentes

Fall 1.1: $vis(\mathbf{v})$ endet in $\mathbf{v}'$

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört zum Sichtbarkeitsgraphen
- $vis(\mathbf{v}) = vis(\mathbf{v}')$

Fall 1.2: $vis(\mathbf{v})$ endet nicht in $\mathbf{v}'$

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört zum Sichtbarkeitsgraphen
- $vis(\mathbf{v}) = segment(\mathbf{v}')$ mit $\mathbf{v}$ ist nicht Endpunkt von $segment(\mathbf{v}')$

Fall 2: Von $\mathbf{v}$ aus gesehen liegt $\mathbf{v}'$ näher als $vis(\mathbf{v})$

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört zum Sichtbarkeitsgraphen
- $vis(\mathbf{v}) = segment(\mathbf{v}')$, das mit $segment(\mathbf{v},\mathbf{v}')$ den kleinsten Winkel im Uhrzeigersinn einschließt

Fall 3: In $\mathbf{v}'$ endet $vis(\mathbf{v})$

Fall 3.1: In $\mathbf{v}'$ endet und beginnt ein Segment

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört zum Sichtbarkeitsgraphen
- $vis(\mathbf{v}) = segment(\mathbf{v}')$, das neu startet

Fall 3.2: In $\mathbf{v}'$ enden zwei Segmente

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört zum Sichtbarkeitsgraphen
- $vis(\mathbf{v}) = vis(\mathbf{v}')$

Fall 4: $\mathbf{v}'$ liegt weiter entfernt als $vis(\mathbf{v})$

- Kante $\{\mathbf{v},\mathbf{v}'\}$ gehört nicht zum Sichtbarkeitsgraphen
- vis bleibt unverändert

Für die Betrachtung von Start- und Endposition der Bewegung werden folgende Transitionen hinzugefügt (vgl. Abbildung 3.9):

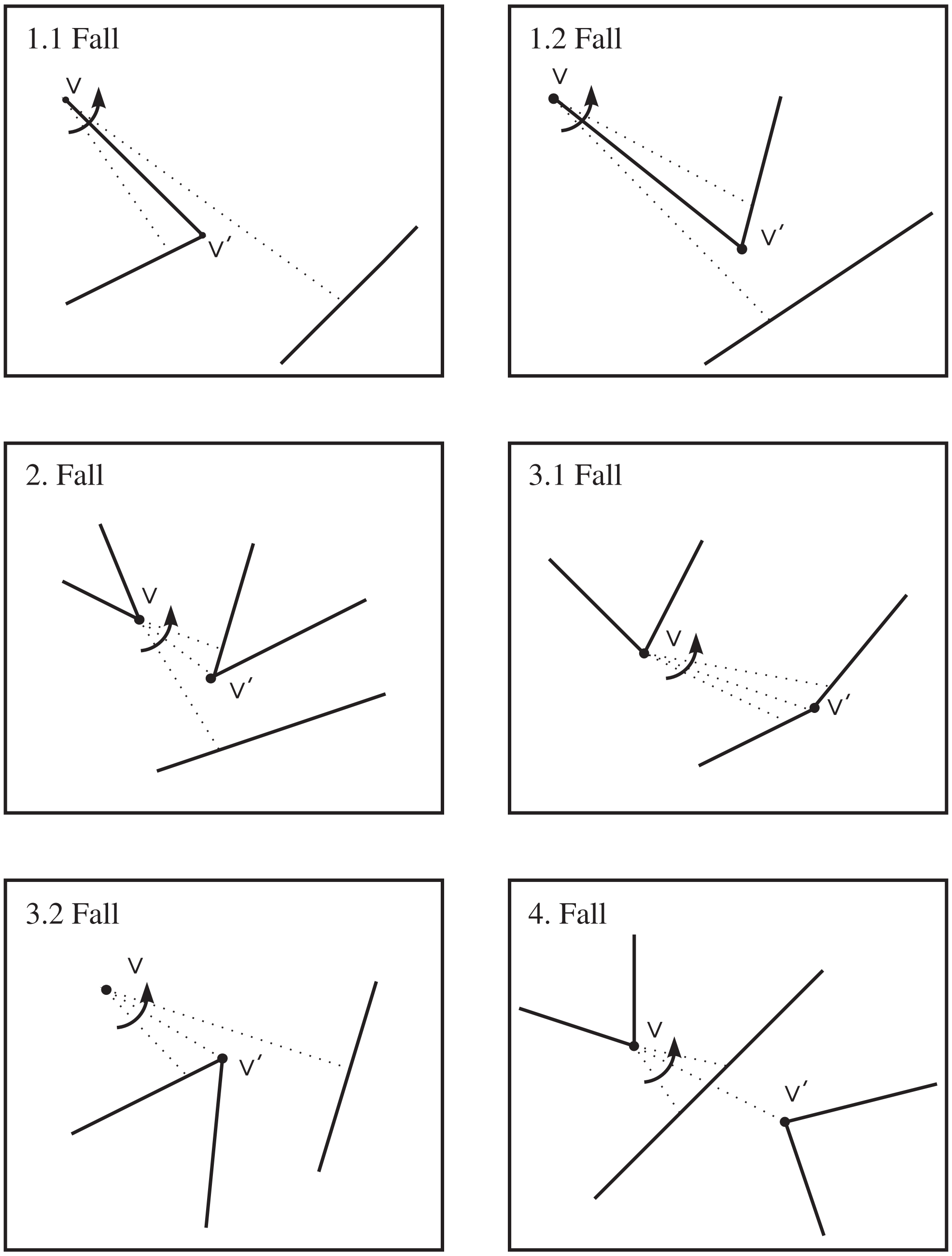

Abbildung 3.8: Transitionen beim Durchlaufen der potentiellen Sichtbarkeitskanten

Fall 1: **p** liegt vor $vis(\mathbf{v})$

- Kante $\{\mathbf{v},\mathbf{p}\}$ gehört zum Sichtbarkeitsgraphen
- vis bleibt unverändert

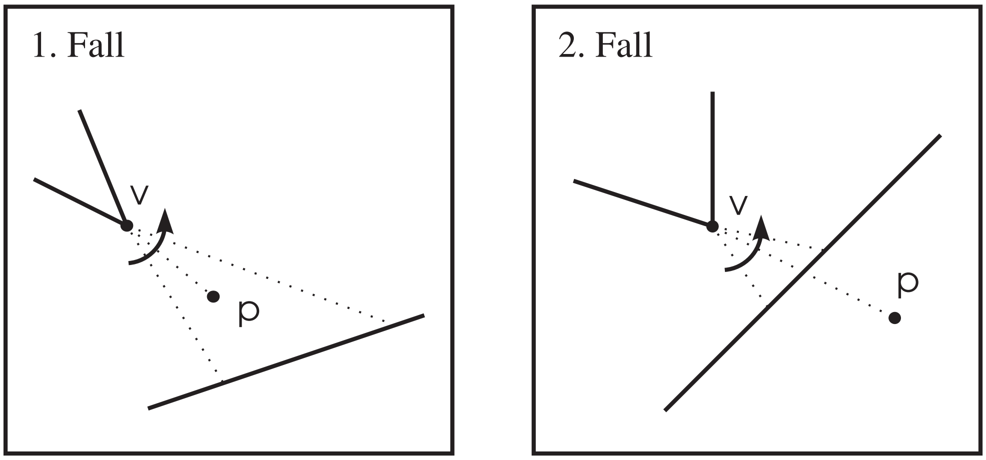

Abbildung 3.9: Transitionen für Punkte

Fall 2: $\mathbf{p}$ liegt hinter $vis(\mathbf{v})$

- Kante $\{\mathbf{v},\mathbf{p}\}$ gehört nicht zum Sichtbarkeitsgraphen
- vis bleibt unverändert

Jede dieser Operationen kann in Zeit $O(1)$ durchgeführt werden. In der Implementierung wurde anstatt der gutartigen Permutation der Sichtbarkeitskanten die Sortierung der Sichtbarkeitskanten nach ihrer Steigung berechnet, so daß sich folgende Gesamtlaufzeit für den implementierten Algorithmus ergibt:

$$\underbrace{O(n^2)}_{\substack{\text{Berechnung der}\\ \text{potentiellen}\\ \text{Sichtbarkeitskanten}}} + \underbrace{O(n^2 \cdot \log n)}_{\substack{\text{Sortierung der potentiellen}\\ \text{Sichtbarkeitskanten}}} + \underbrace{O(n^2)}_{\substack{\text{Test aller potentiellen}\\ \text{Sichtbarkeitskanten}}}$$

$$= O(n^2 \cdot \log n)$$

Dadurch ist folgendes Lemma gezeigt:

Lemma 3.11 :

Gegeben sei eine Menge von k Liniensegmenten, die sich höchstens in den Endpunkten schneiden, und eine Menge von m Punkten ($n = k + m$). Dann hat der Sichtbarkeitsgraph worst-case Größe $O(n^2)$ und kann in Zeit $O(n^2)$ berechnet werden.

3.2.2.2 Bewertung der Kanten

Jeder Kante im Sichtbarkeitsgraphen entspricht eine Translation des bewegten Objektes im Objektraum. Daher wird jede Kante des Sichtbarkeitsgraphen mit der Länge der Translation als Gewicht g_e versehen. Dies kann für jede Kante e mit Endpunkten e_{source} und e_{target} durch die Gleichungen

$$\begin{aligned}\mathbf{t}_e &= (e_{target}.x - e_{source}.x)\mathbf{d}_1 + (e_{target}.y - e_{source}.y)\mathbf{d}_2 \\ g_e &= |\mathbf{t}_e|\end{aligned}$$

erfolgen. Es entstehen dabei ausschließlich nichtnegative Kantengewichte. Dazu benötigt man für jede Kante Zeit $O(1)$. Dadurch ist folgendes Lemma bewiesen:

Lemma 3.12 :

> *Gegeben sei ein Sichtbarkeitsgraph $G_{vis} = (V_{vis}, E_{vis})$ der Größe n. Dann können die Kantengewichte bei Benutzung der euklidischen Metrik in Zeit $O(n)$ berechnet werden.*

3.2.2.3 Kürzeste Wege im Sichtbarkeitsgraphen

Zur Berechnung eines kürzesten Weges zwischen zwei Knoten eines gewichteten, ungerichteten Graphen mit nichtnegativen Kantengewichten gibt es mit Dijkstras Algorithmus einen Standardalgorithmus, der das Problem in einem Graphen mit $|V|$ Knoten und $|E|$ Kanten in einer Zeit von $O(|V| \cdot \log |V| + |E|)$ löst, wenn die Priority Queue der Knoten mit Hilfe von Fibonacci-Heaps realisiert wird.

Damit ergibt sich eine worst-case Laufzeit bei einem Sichtbarkeitsgraphen mit n Knoten und $O(n^2)$ Kanten von $O(n^2)$.

3.3 Zusammenfassung

Im vorhergehenden Abschnitt wurden zum einen die Vorgehensweise des bestbekannten vollständigen Algorithmus zur Berechnung einer euklidisch kürzesten kollisionsfreien Bewegung eines Polyeders bei Polyedern als Hindernissen unter Berücksichtigung zweier translatorischer Freiheitsgrade und zum anderen die Vorgehensweise bei der Implemtierung des Algorithmus im Rahmen dieser Arbeit innerhalb des IGOR -Systems beschrieben.

Dabei wurde auf praktische Aspekte bei der Implementierung eingegangen, die die Verwendung einfacherer Algorithmen rechtfertigt, da im Bereich der algorithmischen Geometrie insbesondere numerische Probleme bei Verwendung von floating-point Arithmetik auftreten. Stabilitätsaspekte bei Algorithmen der algorithmischen Geometrie werden in der neueren Zeit häufiger betrachtet wie z.B. in [MN94].

In der Folge soll nun noch die Gesamtlaufzeit des theoretisch schnellsten und des praktisch implementierten Algorithmus zusammengefaßt werden.

Dabei wird folgende Eingabe betrachtet:

P_{bewegt}	Bewegtes Polyeder der Größe $\lvert P_{bewegt}\rvert$
M_{fest}	Menge von festen Polyedern der Größe $\sum_{P_{fest}\in M_{fest}} \lvert P_{fest}\rvert$
$\mathbf{p}_{start}$	Startposition von P_{bewegt}
$\mathbf{p}_{end}$	Endposition von P_{bewegt}
$\mathbf{d}_1, \mathbf{d}_2$	Richtungsvektoren der Translationsebene für P_{bewegt}

3.3.1 Theoretische worst-case Laufzeit

Sei $n = |P_{bewegt}| + \sum_{P_{fest}\in M_{fest}} |P_{fest}|$

Algorithmus	Eingabegröße	Laufzeit	Ausgabegröße
Kontaktsegmente	n	$O(n^2)$	$O(n^2)$
$\mathrm{ZHK}^{FP}_{p_{start}}$	$O(n^2)$	$O(n^2 \cdot \alpha(n^2) \cdot \log^2(n^2))$ $= O(n^2 \cdot \alpha(n) \cdot \log^2(n))$	$O(n^2 \cdot \alpha(n^2))$ $= O(n^2 \cdot \alpha(n))$
Kürzester Weg	$O(n^2 \cdot \alpha(n))$	$O(n^2 \cdot \alpha(n) \cdot \log^2(n^2 \cdot \alpha(n)))$ $= O(n^2 \cdot \alpha(n) \cdot \log^2(n \cdot \alpha(n)))$	$O(n^2 \cdot \alpha(n))$
Gesamt	n	$O(n^2 \cdot \alpha(n) \cdot \log^2(n \cdot \alpha(n)))$	$O(n^2 \cdot \alpha(n))$

Für den Fall, daß das bewegte Objekt konstante Komplexität hat, ergibt sich

$$\begin{aligned} n &= \sum_{P_{fest}\in M_{fest}} |P_{fest}| \\ k &= |P_{bewegt}| = konstant \end{aligned}$$

Algorithmus	Eingabegröße	Laufzeit	Ausgabegröße
Kontaktsegmente	$n + k$	$O(n)$	$O(n)$
$\mathrm{ZHK}^{FP}_{p_{start}}$	$O(n)$	$O(n \cdot \alpha(n) \cdot \log^2(n))$	$O(n \cdot \alpha(n))$
Kürzester Weg	$O(n \cdot \alpha(n))$	$O(n \cdot \alpha(n) \cdot \log^2(n \cdot \alpha(n)))$	$O(n \cdot \alpha(n))$
Gesamt	$n + k$	$O(n \cdot \alpha(n) \cdot \log^2(n \cdot \alpha(n)))$	$O(n \cdot \alpha(n))$

3.3.2 worst-case Laufzeit des implementierten Algorithmus

Sei $n = |P_{bewegt}| + \sum_{P_{fest}\in M_{fest}} |P_{fest}|$

Algorithmus	Eingabegröße	Laufzeit	Ausgabegröße
Kontaktsegmente	n	$O(n^2)$	$O(n^2)$
$\mathrm{ZHK}^{FP}_{p_{start}}$	$O(n^2)$	$O(n^4 \cdot \log(n^2))$ $= O(n^4 \cdot \log(n))$	$O(n^2 \cdot \alpha(n^2))$ $= O(n^2 \cdot \alpha(n))$
Sichtbarkeitsgraph	$O(n^2 \cdot \alpha(n))$	$O(n^4 \cdot \alpha^2(n) \cdot \log(n^2 \cdot \alpha(n)))$ $= O(n^4 \cdot \alpha^2(n) \cdot \log(n \cdot \alpha(n)))$	$O(n^4 \cdot \alpha^2(n))$
Kürzester Weg	$O(n^4 \cdot \alpha^2(n))$	$O(n^4 \cdot \alpha^2(n))$	$O(n^2 \cdot \alpha(n))$
Gesamt	n	$O(n^4 \cdot \alpha^2(n) \cdot \log(n \cdot \alpha(n)))$	$O(n^2 \cdot \alpha(n))$

Für den Fall, daß das bewegte Objekt konstante Komplexität hat ergibt sich

$$\begin{aligned} n &= \sum_{P_{fest} \in M_{fest}} |P_{fest}| \\ k &= |P_{bewegt}| = konstant \end{aligned}$$

Algorithmus	Eingabegröße	Laufzeit	Ausgabegröße
Kontaktsegmente	$n + k$	$O(n)$	$O(n)$
$\text{ZHK}_{p_{start}}^{FP}$	$O(n)$	$O(n^2 \cdot \log(n))$	$O(n \cdot \alpha(n))$
Sichtbarkeitsgraph	$O(n \cdot \alpha(n))$	$O(n^2 \cdot \alpha^2(n) \cdot \log(n \cdot \alpha(n)))$	$O(n^2 \cdot \alpha^2(n))$
Kürzester Weg	$O(n^2 \cdot \alpha^2(n))$	$O(n^2 \cdot \alpha^2(n))$	$O(n \cdot \alpha(n))$
Gesamt	$n + k$	$O(n^2 \cdot \alpha^2(n) \cdot \log(n \cdot \alpha(n)))$	$O(n \cdot \alpha(n))$

3.4 Erweiterung um die hierarchisch inkrementelle Strategie

Im Rahmen der Diplomarbeit von M. Fritz [Fr93] wurde im IGOR-System die in *Kapitel 2* angesprochene Kollisionserkennung für Translationen und Rotationen von Polyedern implementiert. Mit Hilfe dieser Funktionalität wurde der Kollisionserkennungsteil der heuristisch inkrementellen Strategie für den obigen Bewegungsplanungsalgorithmus implementiert. Dabei wurden im iterativen Prozeß die zusätzlichen Vorteile bei der Verwendung des Konfigurationsraumansatzes genutzt.

3.4.1 Erweiterung des Konfigurationsraumansatzes

Um bei den Iterationen der heuristisch inkrementellen Vorgehensweise nicht das Aussehen der gesamten Zusammenhangskomponente des Freiraums des Konfigurationsraumes, die die Startposition der Bewegung beinhaltet, neu berechnen zu müssen, wird die Beschreibung der in der vorhergehenden Iteration berechneten Zusammenhangskomponente wiederverwendet.

Dies ist unmittelbar möglich, da für die Folge der betrachteten Hindernismengen gilt:

$$H_i \subset H_{i+1}$$

Für die neu hinzukommenden Hindernisse

$$H_{i+1}^{neu} = H_{i+1} \setminus H_i$$

werden die Kontaktsegmente berechnet und die zu den Kanten von ZHK_i^{FP} gehörenden hinzugefügt. Ausgehend von dieser Gesamtmenge von Kontaktsegmenten wird das Aussehen von ZHK_{i+1}^{FP} berechnet. Da i.a. nur eine geringe Anzahl von Kontaktsegmenten das Aussehen der zu berechnenden Zusammenhangskomponente bestimmen, wird durch die Benutzung von ZHK_i^{FP} die Anzahl der zu betrachtenden Liniensegmente bei der Berechnung von ZHK_{i+1}^{FP} verringert und damit die Berechnung gegenüber einer vollständig neuen Betrachtung der Szene beschleunigt.

3.4.2 Interaktive Spezifikation der Ausgangsszene

Zusätzlich können in der betrachteten Bewegungsplanungsszene Hindernisse spezifiziert werden, die in die betrachtete Anfangszene B_0 eingefügt werden. Dies bedeutet keine Abweichung von der allgemeinen Strategie, sondern eine Erweiterung um eine interaktive Komponente, die den Suchprozess der heuristisch inkrementellen Vorgehensweise durch Benutzereingaben zusätzlich steuert.

Durch die interaktive Spezifikation der Ausgangsszene kann die Heuristik dazu gebracht werden, schon im ersten Schritt einen Weg zu berechnen, der durch darauffolgende Iterationen nur noch lokale Änderungen erhält. Damit kann die Laufzeit für die Bewegungsplanung durch Interaktion mit dem Benutzer verkürzt werden.

3.5 Praktische Laufzeitergebnisse

In *Anhang A* werden Laufzeitergebnisse für zwei Szenen beschrieben, die beispielhaft die Stärken und Schwächen der heuristisch inkrementellen Bewegungsplanungsstrategie im Zusammenhang mit dem klassischen Bewegungsplanungsalgorithmus dieses Kapitels aufzeigen.

Dabei wurden die beiden Varianten der heuristisch inkrementellen Bewegungsplanungsstrategie betrachtet, bei der alle kollidierenden Hindernisse in die nächste Iteration der Bewegungsplanung übernommen werden *(inkrementell)* und bei der nur das erste kollidierende Hindernis berücksichtigt wird *(minimal inkrementell)*.

Kapitel 4

Heuristische Bewegungsplanung mit bewegbaren Hindernissen im $\mathbb{R}^3$

In *Kapitel 2* wurde eine heuristische Bewegungsplanungsstrategie beschrieben, deren Vorgehensweise sich an der eines menschlichen Planers orientiert. Daraus ergab sich eine heuristisch inkrementelle Strategie, mit deren Hilfe bekannte Bewegungsplanungsalgorithmen beschleunigt werden können, wobei die Menge der dabei von diesen Algorithmen gelösten Bewegungsplanungsprobleme höchstens vergrößert wird.

Das in Abbildung 4.1 dargestellte Problem ist mit Hilfe klassischer Bewegungsplanungsalgorithmen nicht lösbar; jedoch stellt es für einen menschlichen Planer kein intuitiv schweres Bewegungsplanungsproblem dar, denn es existiert als Lösung die einfache Bewegungsfolge, die in Abbildung 4.2 dargestellt ist. Dies resultiert aus der Tatsache, daß ein menschlicher Planer in seiner Strategie die Objekte in einer Szene nicht wie die meisten Bewegungsplanungsalgorithmen a priori in die zwei Mengen *bewegte Objekte* und *Hindernisse* einteilt, sondern die Menge der *Hindernisse* unterteilt in die zwei Mengen *bewegbare Hindernisse* und *unbewegliche Hindernisse*. Mit Hilfe dieser verfeinerten Einteilung der Objekte in der Szene ist eine Modifikation der Bewegungsplanung möglich. Der menschliche Bewegungsplaner versucht nun das/die bewegte(n) Objekt(e) in der Szene zu bewegen und bei Bedarf werden die bewegbaren Hindernisse zu diesem Zweck aus ihrer Ausgangslage verschoben, um die Bewegung(en) des/der bewegten Objekte(s) zu ermöglichen.

Dabei kann in der Regel beobachtet werden, daß für die bewegbaren Hindernisse, die aus dem Weg geräumt werden, möglichst kurze Bewegungen gewählt werden, da der menschliche Planer in der Regel nur nach lokalen Ausweichbewegungen sucht. Theoretisch ist die Nachahmung einer solchen Vorgehensweise exakt durch die Berechnung eines Konfigurationsraumes für alle bewegten Objekte und bewegbaren Hindernisse in einer Szene möglich. Da jedoch i.a. viele bewegbare Hindernisse in einer Szene vorhanden sind, ist die Berechnung eines solchen Konfigurationsraumes, wie es mit dem in [SS83b] beschriebenen Algorithmus möglich ist, in der Praxis nicht verwendbar. Gesucht ist demnach eine Vorgehensweise, die heuristisch die Strategie eines menschlichen Planers als Vorbild nimmt und damit versucht, die Lösungsmöglichkeiten klassischer Bewegungsplanungsalgorithmen zu erweitern.

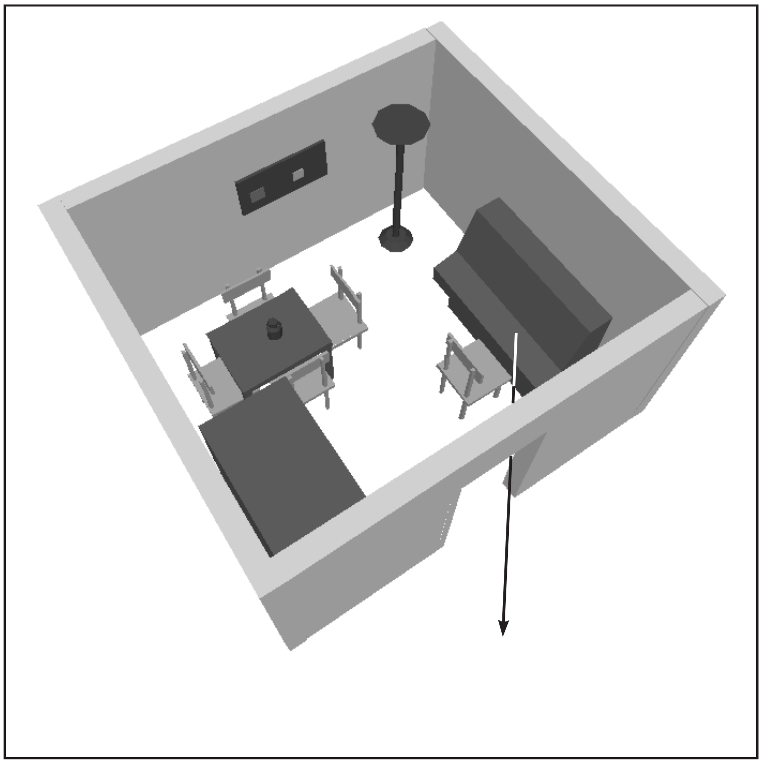

Abbildung 4.1: Bewegungsplanungsproblem ohne klassische Lösung

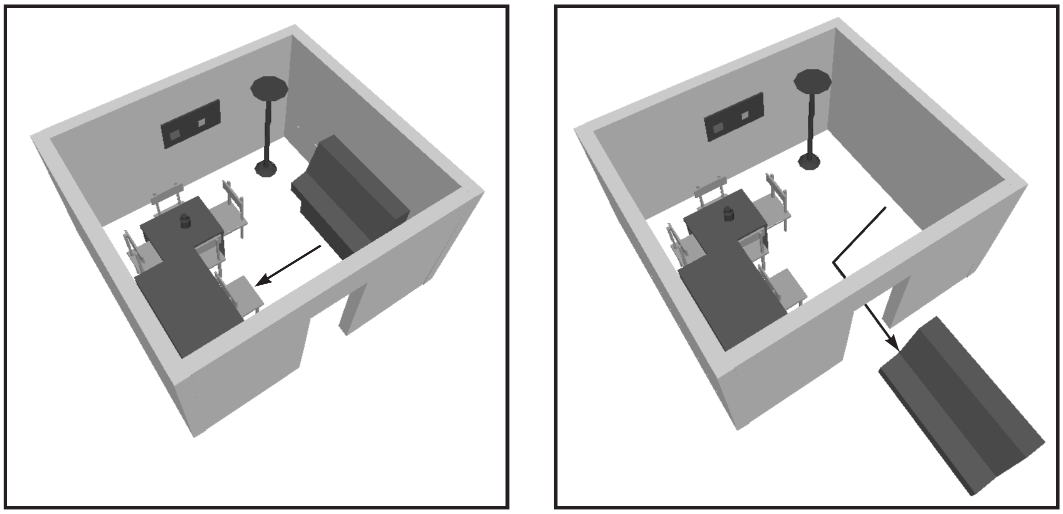

Abbildung 4.2: Lösung des Bewegungsplanungsproblems ohne klassische Lösung

Mit dem Problem bewegbarer Hindernisse in einer Szene haben sich zuvor Wilfong [Wi91] und Dhagat und O'Rourke [DO92] beschäftigt. Betrachtet wurden Szenen im $\mathbb{R}^2$, bei denen ein Roboter entweder alleine bewegt wird, ein anderes Objekt zieht oder ein bzw. mehrere Objekt(e) verschiebt. Bewegungen für die bewegbaren Hindernisse kommen dabei nur durch die Bewegungen des Roboters zustande.

In Anlehnung an die Vorgehensweise eines menschlichen Planers soll bei unserer Vorgehensweise nicht das Greifen, Ziehen oder Verschieben der Objekte im Mittelpunkt stehen, sondern das Erkennen der Notwendigkeit, daß Hindernisse beseitigt werden müssen, sowie die Berechnung und die Koordination der Bewegungen der Hindernisse. Dadurch sollen die Möglichkeiten der klassischen Bewegungsplanungsalgorithmen erweitert werden.

Grundsätzlich wird die Betrachtung der klassischen Bewegungsplanungsalgorithmen, die die Objekte in einer Szene in die beiden Kategorien

1. bewegtes Objekt
2. Hindernisse

aufteilen, erweitert, indem die zweite Kategorie *Hindernisse* aufgeteilt wird in die Kategorien

1. bewegbare Hindernisse
2. feste Hindernisse

so daß drei Kategorien von Objekten in einer Szene enthalten sind:

1. bewegtes Objekt
2. bewegbare Hindernisse
3. feste Hindernisse

Dabei ergibt sich durch die Definition von bewegbaren Hindernissen die Möglichkeit für die Heuristik, die Lösungsmöglichkeiten der klassischen Bewegungsplanungsalgorithmen zu erweitern.

Ziel dabei ist es, auf der einen Seite die Menge aller Hindernisse zu berechnen, die bewegt werden müssen, um ein Bewegungsplanungsproblem zu lösen, und auf der anderen Seite die dazu nötigen Bewegungen für die Objekte zu berechnen und zu koordinieren.

Formal ist dabei folgendes Problem zu lösen:

gegeben:

P_{bewegt}	bewegtes Objekt
$H_{bewegbar}$	Menge bewegbarer Hindernisse
H_{fest}	Menge fester Hindernisse
p_{start}	Startposition der Bewegung für P_{bewegt}
p_{end}	Endposition der Bewegung für P_{bewegt}

gesucht:

Gibt es Bewegungen w_{bewegt} für das bewegte Objekt und $w^i_{bewegbar}$ für bewegbare Hindernisse $h_i \in H_{bewegbar}$, so daß P_{bewegt} durch w_{bewegt} kollisionsfrei von p_{start} nach p_{end} bewegt werden kann, nachdem alle h_i ihre Bewegungen $w^i_{bewegbar}$ ausgeführt haben?

Die Idee dabei ist, aus der Menge der bewegbaren Hindernisse die Menge der Hindernisse zu berechnen, durch deren Entfernen aus der Ausgangsposition eine kollisionsfreie Bewegung für das bewegte Objekt ermöglicht wird. Da diese Menge i.a. gegenüber der Menge aller bewegbaren Hindernisse klein ist und die weitere Betrachtung von Bewegungen für Hindernisse auf diese Teilmenge eingeschränkt werden kann, wird eine Verringerung der Komplexität des zu lösenden Bewegungsplanungsproblems gegenüber der Betrachtung aller bewegbaren Hindernisse erreicht.

Dazu wird für das bewegte Objekt P_{bewegt} nur unter Berücksichtigung der festen Hindernisse ein kollisionsfreier Weg von p_{start} nach p_{end} berechnet und dieser auf Kollisionen mit bewegbaren Hindernissen getestet. Liegen keine Kollisionen mit bewegbaren Hindernissen vor, ist der geplante Weg kollisionsfrei bzgl. aller Hindernisse in der Szene und die Bewegungsplanung ist abgeschlossen. Liegen Kollisionen mit einer Teilmenge $H^{col}_{bewegbar} \subset H_{bewegbar}$ von bewegbaren Hindernissen vor, wird die Menge der während der Lösung des gegebenen Bewegungsplanungsproblems bewegten Objekte auf die Menge $\{P_{bewegt}\} \cup H^{col}_{bewegbar}$ beschränkt.

Für diese Menge könnte nun mit Hilfe eines vollständigen Verfahrens eine Lösung für das Bewegungsplanungsproblem berechnet werden, jedoch ist auch diese Vorgehensweise wegen der Anzahl der zu berücksichtigenden Freiheitsgrade der beteiligten bewegten Objekte praktisch nicht anwendbar.

Um das verbleibende Bewegungsplanungsproblem mit i.a. reduzierter Anzahl von beweglichen Objekten mit Hilfe eines praktisch anwendbaren Verfahrens zu lösen, ist eine weitere Aufteilung des verbleibenden, komplexen Problems in eine Folge von einfacheren, praktisch lösbaren Teilproblemen notwendig. Dies erreicht man, indem für jedes der beweglichen Objekte separat eine Bewegung berechnet wird und diese Bewegungen in einem anschließenden Schritt koordiniert werden.

Dabei geht gegenüber der exakten Vorgehensweise zur Berechnung der Lösung solcher Bewegungsplanungsprobleme die Vollständigkeit des Algorithmus verloren, jedoch erhält man dadurch einen Algorithmus zur praktischen Berechnung von Lösungen dieser Klasse von Bewegungsplanungsproblemen.

Bei der Zerlegung des Gesamtproblems in mehrere praktisch lösbare Teilprobleme orientiert sich die heuristische Vorgehensweise wieder an der Strategie eines menschlichen Planers. Dieser führt die Klassifizierung der Hindernisse in einer Szene a priori in *bewegbare Hindernisse* und *feste Hindernisse* durch und betrachtet das Bewegungsplanungsproblem für das bewegte Objekt implizit ohne die bewegbaren Hindernisse. Dafür berechnet er eine Lösung und testet die Lösung auf Kollisionen mit bewegbaren Hindernissen. Für alle störenden, bewegbaren Hindernisse werden dann sukzessive Ausweichbewegungen berechnet und so koordiniert, daß diese ausgeführt werden können.

Diese Vorgehensweise kann folgendermaßen algorithmisch beschrieben werden:

P	bewegtes Objekt
BPA	Bewegungsplanungsalgorithmus zur Berechnung der Bewegung für P
p_{start}	Startposition von P
p_{end}	Endposition von P
w	Weg für das bewegte Objekt
H_b	Menge bewegbarer Hindernisse
h_b	bewegbares Hindernis in H_b
p_{h_b}	Position des bewegbaren Hindernis h_b
H_b^{col}	Teilmenge der bewegbaren Hindernisse, die w für P behindern
BPA_{h_b}	Bewegungsplanungsalgorithmus für $h_b \in H_b$
M_{w_b}	Menge von Ausweichbewegungen für die Hindernisse aus H_b^{col}
H_f	Menge fester Hindernisse

$w \leftarrow BPA(P, H_f, p_{start}, p_{end})$
$H_b^{col} \leftarrow Kollision(P, H_b, w)$
if $(H_b^{col} \neq \emptyset)$ **then**
    $M_{w_b} \leftarrow \emptyset$
    forall $h_b \in H_b^{col}$ **do**
        $w_b \leftarrow BPA_{h_b}(h_b, H_f \cup (H_b \setminus H_b^{col}), P, w, p_{h_b})$
        $M_{w_b} \leftarrow M_{w_b} \cup \{(h_b, w_b)\}$
    $KoordiniereBewegungen(M_{w_b})$

Für die Berechnung der Bewegung für das bewegte Objekt kann einer der bekannten klassischen Bewegungsplanungsalgorithmen verwendet werden, insbesondere auch unter Verwendung der in *Kapitel 2* beschriebenen heuristisch inkrementellen Bewegungsplanungsstrategie.

Auf die Kollisionserkennung wurde bereits im Zusammenhang mit der heuristisch inkrementellen Vorgehensweise eingegangen, so daß an dieser Stelle auf die Ausführungen der Seiten 12 – 34 verwiesen wird.

Eine nähere Betrachtung ist somit noch für die beiden Teilprobleme

1. Berechnung der Ausweichbewegungen für bewegbare Hindernisse

2. Koordination von berechneten Ausweichbewegungen

nötig.

Die beiden Teilprobleme sind getrennt oder in einem gemeinsamen Schritt lösbar. Beide Vorgehensweisen werden in der Folge näher betrachtet.

Das erste Problem bei der Berechnung der Ausweichbewegung für jedes einzelne störende, bewegbare Hindernis ist die Bestimmung des Endpunktes der Ausweichbewegung, da der Endpunkt nicht durch die Problemstellung gegeben ist.

4.1 Bestimmung der möglichen Endpunkte von Ausweichbewegungen

Für ein bewegbares Hindernis, für das eine Ausweichbewegung berechnet werden soll, muß in einem ersten Schritt der Endpunkt oder eine Menge von potentiellen Endpunkten der Bewegung berechnet werden. Das zu lösende Problem läßt sich folgendermaßen charakterisieren:

gegeben:

$h_{bewegbar}$	bewegbares Hindernis
P	bewegtes Objekt
w	berechneter Weg des bewegten Objektes
H_f	Menge der als fest angesehenen Hindernisse
$\mathcal{P}_{h_{bewegbar}}$	Positionsraum von $h_{bewegbar}$

gesucht:

$$P_{end} = \left\{ \mathbf{p}_{end} \in \mathcal{P}_{h_{bewegbar}} \mid \left(P(w(t)) \cup \left(\bigcup_{h_f \in H_f} h_f \right) \right) \cap int(h_{bewegbar}(\mathbf{p}_{end})) = \emptyset,\ \forall\ t \in [\,0,1] \right\}$$

Anschaulich ist die Menge aller Positionen des bewegbaren Hindernisses $h_{bewegbar}$ zu berechnen, in denen das Innere von $h_{bewegbar}$ einen leeren Schnitt mit P zu jedem Zeitpunkt der Bewegung w hat und gleichzeitig keines der als fest angesehenen Hindernisse schneidet. D.h. zu berechnen ist das Komplement des Konfigurationsraumhindernisses des überstrichenen Volumens von P während w geschnitten mit dem Komplement der Vereinigung der Konfigurationsraumhindernisse der als fest angesehenen Hindernisse. Das entspricht dem Schnitt des Freiraums im Konfigurationsraum von $h_{bewegbar}$ bzgl. des überstrichenen Volumens von P während w mit dem Freiraum im Konfigurationsraum von $h_{bewegbar}$ bzgl. der als fest angesehenen Hindernisse H_f. Diese Beschreibung legt eine Vorgehensweise zur Berechnung von P_{end} nahe. Da es für die Berechnung der möglichen Endpositionen der Ausweichbewegung für $h_{bewegbar}$ sinnvoll ist, einen Konfigurationsraum zu berechnen, werden für die Berechnung der Ausweichbewegungen der bewegbaren Hindernisse in der Folge nur Bewegungsplanungsalgorithmen betrachtet, die auf dem Konfigurationsraumansatz basieren.

Die Berechnung der möglichen Endpositionen des bewegbaren Hindernisses bzgl. des bewegten Objektes P erfolgt analog zur Berechnung der Konfigurationsraumhindernisse für feste Hindernisse in der Bewegungsplanung und als Hindernis wird das überstrichene Volumen von P während w benutzt.

Es bleibt das Problem der Berechnung des überstrichenen Volumens von P während w.

4.1.1 Berechnung des überstrichenen Volumens von Polyedern bei Bewegungen

Die Komplexität der Berechnung eines überstrichenen Volumenkörpers für ein Polyeder hängt von der Art der ausgeführten Bewegung ab. Für die weiteren Berechnungen ist es vorteilhaft, wenn der Volumenkörper selbst wieder ein Polyeder ist, da die meisten Bewegungsplanungsalgorithmen von Polyedern als Objekte ausgehen. Für Translationen von Polyedern kann der

Volumenkörper exakt berechnet werden, da er selbst wieder ein Polyeder ist. Bei Rotationen ergeben sich als Flächen des Volumenkörpers Flächen zweiter Ordnung, so daß durch die Bestimmung eines Volumenpolyeders der exakte Volumenkörper nur approximiert werden kann. Bei allgemeinen Bewegungen entstehen i.a. keine Polyeder als Volumenkörper, so daß ein berechneter Volumenpolyeder den exakten Volumenkörper ebenfalls nur approximieren kann. In der Folge soll nun zunächst für die beiden Spezialfälle von Translation und Rotation und später auch für allgemeine Bewegungen die Berechnung eines Volumenpolyeders erläutert werden.

4.1.1.1 Volumenpolyeder bei translatorischen Bewegungen

Formal läßt sich das zu lösende Problem folgendermaßen zusammenfassen:

gegeben:

P bewegtes Polyeder
$\mathbf{t}$ Translationsvektor für P

gesucht:

$Vol_P^t = \{\mathbf{p} \in \mathbb{R}^3 \mid \exists\ 0 \leq \mu \leq 1 :\ \mathbf{p} \in P(\mu\mathbf{t})\}$

Für den Fall, daß das Polyeder P konvex ist, ergibt sich das Volumenpolyeder als konvexe Hülle der Punktmenge bestehend aus den Ecken des Polyeders in der Ausgangslage und den Ecken des Polyeders nach der Translation. D.h. sei V_P die Eckenmenge des Polyeders P in der Ausgangslage und $V_P(\mathbf{t})$ die Eckenmenge von P nach der Translation, so ergibt sich das Volumenpolyder als

$$Vol_P^t = ConvHull(V_P \cup V_P(\mathbf{t}))$$

Daraus ergibt sich eine direkte Möglichkeit zur Berechnung des Volumenkörpers mit Hilfe eines Algorithmus zur Berechnung konvexer Hüllen von Punktmengen im $\mathbb{R}^3$. Die dabei betrachtete Punktmenge hat die Größe $O(|P|)$, so daß die konvexe Hülle in Zeit $O(|P|\cdot\log|P|)$ berechnet werden kann (vgl. [Ed87]).

Der Volumenkörper für konvexe Polyeder kann jedoch auch mit Hilfe eines Linearzeitalgorithmus folgendermaßen berechnet werden:

Der Translationsvektor $\mathbf{t}$ unterteilt die Flächen des Polyeders P in zwei disjunkte Teilmengen F^+ und F^-, wobei

$$\begin{aligned} F^- &= \{f \in F_P \mid \mathbf{n}_f\mathbf{t} \leq 0\} \\ F^+ &= F_P \setminus F^- \\ &= \{f \in F_P \mid \mathbf{n}_f\mathbf{t} > 0\} \end{aligned}$$

Dabei bedeutet $\mathbf{n}_f\mathbf{t} \leq 0$, daß $\mathbf{n}_f$ und $\mathbf{t}$ einen Winkel aus dem Intervall $[90°, 180°]$ einschließen.

Nach Vereinbarung zeigen die Normalenvektoren der Flächen in das Körperäußere, so daß F^- der Menge der Flächen von P entspricht, die während der Translation nicht durch andere Punkte von P überstrichen werden und F^+ der Menge der Flächen von P, die von P bei einer Translation in Richtung **t** durchlaufen wird. Dadurch ergibt sich aus der Kantenmenge des Polyeders eine zyklische Folge von Kanten E_z, die adjazent sind zu einer Fläche aus F^+ und einer Fläche aus F^-. Diese Kantenfolge definiert einen Schnitt durch die äußere Begrenzung von P, entlang dem P aufgetrennt wird und die Kanten aus E_z zu Flächen des Volumenpolyeders expandiert werden.

Das Volumenpolyeder Vol_P^t ergibt sich dann durch folgende Konstruktion:

1. Berechne die zyklische Kantenfolge E_z
2. Expandiere die Kanten aus E_z zu parallelogrammförmigen Flächen in Richtung der Translation **t** und verbinde die zu Kanten aus E_z adjazenten Flächen von F^+ mit den verschobenen Kanten aus E_z.

Anschaulich betrachtet wird P entlang von E_z aufgetrennt und zwischen die beiden Polyederhälften P_{F^-} und P_{F^+} ein „Flächentorus" aus zu Flächen expandierten Kanten von E_z geschoben.

Diese Vorgehensweise ist in Abbildung 4.3 graphisch veranschaulicht.

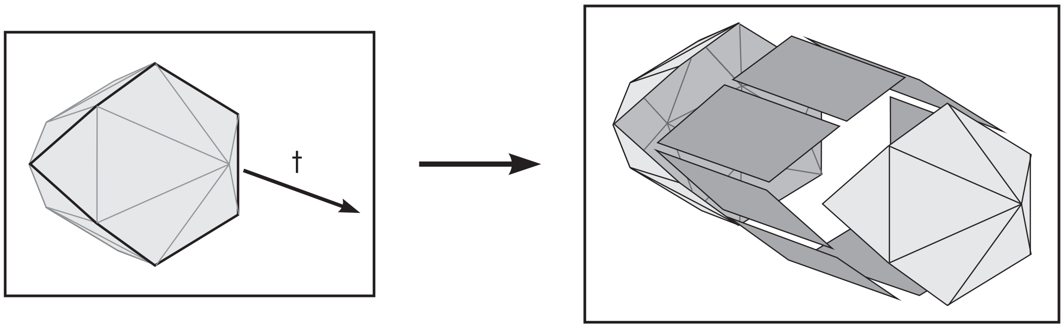

Abbildung 4.3: Berechnung des Volumenpolyeders bei Translationen

Lemma 4.1 :

*Sei P ein konvexes Polyeder, **t** ein Translationsvektor, Vol_P^t der von P während der Translation überstrichene Volumenkörper und $Konst_P^t$ das durch den Linearzeitalgorithmus berechnete Volumenpolyeder. Dann ist $Konst_P^t = Vol_P^t$.*

Beweis:
Die zyklische Kantenfolge E_z definiert nur einen Schnitt durch die äußere Begrenzung von P und keinen eindeutigen Schnitt von P in zwei Polyeder. Einer der möglichen Schnitte teilt P in folgende Punktmengen:

$$\begin{aligned} P_{hinten} &= P \setminus F^+ \\ P_{vorne} &= F^+ \end{aligned}$$

Dadurch ergibt sich $Konst_P^t$ mengentheoretisch als Vereinigung der folgenden Mengen:

$$Konst_P^t = P_{vorne} \cup \mathrm{Prisma}_{\mathbf{t}}(F^+) = P \cup \mathrm{Prisma}_{\mathbf{t}}(F^+)$$

wobei $\mathrm{Prisma}_{\mathbf{t}}(F^+)$ das Polyeder ist, das sich durch Verschiebung der Flächen aus F^+ um $\mathbf{t}$ ergibt.

1. $Vol_P^t \subseteq Konst_P^t$
   $P(0)$ sei das Polyeder P in seiner Ausgangsposition, $P(\mathbf{v})$ das um den Vektor $\mathbf{v}$ verschobene Polyeder P und $\mathbf{p} \in Vol_P^t$ beliebig.

1. Fall: $\mathbf{p} \in P(0)$
   $\Rightarrow \mathbf{p} \in Konst_P^t$ wegen der mengentheoretischen Betrachtung von $Konst_P^t$

2. Fall: $\mathbf{p} \in Vol_P^t \setminus P(0)$
   $\Rightarrow\ \exists\, 0 \leq \alpha \leq 1,\ \mathbf{p} \in P(\alpha \mathbf{t})$
   Für die Projektion auf eine Ebene senkrecht zur Translationsebene gilt nach Konstruktion:

   $$proj_{\mathbf{t}}(F^+) = proj_{\mathbf{t}}(P)$$

   D.h. $\exists\, f \in F^+,\ \mathbf{q} \in f:\ proj_{\mathbf{t}}(\mathbf{q}) = proj_{\mathbf{t}}(\mathbf{p})$
   Sei $\mathbf{v_p} \in P(0)$ mit $\mathbf{p} = \mathbf{v_p}(\alpha \mathbf{t})$
   Nach Konstruktion von F^+ und da $\mathbf{p} \notin P(0)$ gilt für $\mathbf{v_p}$:

   $$\exists\, 0 \leq \beta \leq \alpha:\ \mathbf{v_p}(\beta \mathbf{t}) = \mathbf{q}$$

   $\Rightarrow \mathbf{q}((\alpha - \beta)\mathbf{t}) = \mathbf{p}$
   $\Rightarrow \mathbf{p} \in \mathrm{Prisma}_{\mathbf{t}}(F^+)$
   $\Rightarrow \mathbf{p} \in Konst_P^t$

2. $Konst_P^t \subseteq Vol_P^t$
   Sei $\mathbf{p} \in Konst_P^t$ beliebig.

1. Fall: $\mathbf{p} \in P\ \Rightarrow \mathbf{p} \in Vol_P^t$

2. Fall: $\mathbf{p} \in \mathrm{Prisma}_{\mathbf{t}}(F^+)$
   Sei $\mathbf{q} \in f,\ f \in F^+$ mit $proj_{\mathbf{t}}(\mathbf{p}) = proj_{\mathbf{t}}(\mathbf{q})$.
   $\Rightarrow\ \exists\, 0 \leq \alpha \leq 1:\ \mathbf{q}(\alpha \mathbf{t}) = \mathbf{p}$
   $\Rightarrow \mathbf{p} \in Vol_P^t$ nach Definition von Vol_P^t

■

Als Laufzeit für den Algorithmus ergibt sich:

$$\underbrace{O(n)}_{\substack{\text{Berechnung der}\\ \text{Kantenfolge } E_z}} + \underbrace{O(n)}_{\substack{\text{Expansion der Kanten}\\ \text{aus } E_z}}$$

$$= \quad O(n)$$

Ist ein Körper nicht konvex, so kann der Volumenkörper durch den Volumenkörper seiner konvexen Hülle approximiert werden. Zur exakten Berechnung des Volumenkörpers zerlegt man das Polyeder in konvexe Teile und berechnet für diese die Volumenkörper. Durch die Triangulierung des Polyeders erhält man eine Menge von $O(|P|^2)$ Tetraedern, die in Zeit $O(|P|^2 \cdot \log|P|)$ berechnet werden können (vgl. [CP90]). Für diese haben die Volumenpolyeder wieder konstante Komplexität und lassen sich in konstanter Zeit berechnen.

Nach [AS93] kann die Vereinigung einer Menge von k konvexen Polyedern mit n Flächen in erwarteter Zeit $O(k^3 + k \cdot n \cdot \log^3 k)$ berechnet werden. Für den Fall von k konvexen Polyedern konstanter Komplexität gilt für diese Vereinigung, daß sie nicht komplexer sein kann, als die Beschreibung des Arrangements der $O(k)$ Hyperebenen, die die k Polyeder beschreiben. Nach [Ed87] hat dieses Arrangement im $\mathbb{R}^3$ höchstens Komplexität $O(k^3)$, so daß $O(k^3)$ eine obere Schranke für die Komplexität des Volumenpolyeders darstellt.

Für nichtkonvexe Polyeder kann man sich nun die Berechnung der Begrenzung des Volumenpolyeders auf die Betrachtung der einzelnen Flächen zurückgeführt vorstellen. Dabei berechnet man die Flächen des Volumenpolyeders, die entstehen, wenn eine Fläche von P um $\mathbf{t}$ verschoben wird. Dadurch erhält man für jede Fläche f des Polyeders einen Volumenpolyeder der Größe $O(|f|)$, dessen Flächen zur Beschreibung des gesamten Volumenpolyeders beitragen können. Man kann o.B.d.A. annehmen, daß die Flächen von P trianguliert sind, so daß man eine Menge von $O(|P|)$ Volumenpolyedern konstanter Größe erhält, deren Flächen die gesamte Begrenzung des Volumenpolyeders beschreiben. Die mit Hilfe dieser Flächen beschriebene Polyederbegrenzung kann höchstens Komplexität $O(|P|^3)$ haben, so daß der gesamte Volumenpolyeder auch eine Komplexität von $O(|P|^3)$ hat.

Dadurch wird folgende Aussage gezeigt:

Lemma 4.2 :

> *Gegeben sei ein Polyeder P und ein Translationsvektor $\mathbf{t}$. Sei Vol_P^t das Polyeder, das das von P während der Translation $\mathbf{t}$ überstrichene Volumen beschreibt.*
>
> 1. *Ist P konvex, kann Vol_P^t in Zeit $O(|P|)$ berechnet werden und hat Komplexität $O(|P|)$.*
> 2. *Ist P nicht konvex und eine Zerlegung in k konvexe Teile bekannt, kann eine Menge von k konvexen Polyedern, deren Vereinigung Vol_P^t bildet, in Zeit $O(k \cdot |P|)$ berechnet werden und die Vereinigung hat Komplexität $O(k^3 + k \cdot |P| \cdot \log^2 k)$.*
> 3. *Ist P nicht konvex, kann eine Menge von $O(|P|^2)$ konvexen Polyedern, deren Vereinigung Vol_P^t bildet, in Zeit $O(|P|^2 \cdot \log|P|)$ berechnet werden und die Vereinigung hat Komplexität $O(|P|^3)$.*

4.1.1.2 Volumenpolyeder bei rotatorischen Bewegungen

Analog zur Translation ist nun folgendes Problem zu lösen:

gegeben:

P	Polyeder
$(\mathbf{r}, \theta)$	Rotation um normierte Rotationsachse $\mathbf{r}$ um Winkel θ

gesucht:

$$Vol_P^{(\mathbf{r},\theta)} = \{\mathbf{p} \in \mathbb{R}^3 \mid \exists\ 0 \leq \alpha \leq \theta,\ \mathbf{p} \in P(\mathbf{r}, \alpha)\}$$

Bei rotatorischen Bewegungen entstehen Volumenkörper, die nicht exakt als Polyeder beschreibbar sind. Die Berechnung eines Volumenpolyeders ist daher immer eine Approximation des tatsächlichen Volumenkörpers.

Ist P ein nichtkonvexes Polyeder, betrachtet man dessen Zerlegung in konvexe Teile und berechnet für diese eine Menge von Polyedern, deren Vereinigung die Approximation des Volumenkörpers bilden.

Die Idee dabei ist, daß man für jede Ecke des Polyeders ein Polygon berechnet, in dem sich die Ecke während der Rotation bewegt. Berechnet man danach von der Menge der Eckpunkte der Polygone die konvexe Hülle, erhält man eine Approximation des Volumenkörpers durch ein konvexes Polyeder.

O.B.d.A. verlaufe die Rotationsachse in Richtung der z-Achse, so daß die Eckpunkte des Polyeders in Ebenen parallel zur x-y-Ebene rotieren. Dadurch kann die Betrachtung der einzelnen Eckpunkte auf die Betrachtung in einer Ebene unter Berücksichtigung nur der x- und y-Koordinaten der Eckpunkte zurückgeführt werden, wobei die Rotation im $\mathbb{R}^3$ auf die Rotation in der Ebene um den Schnittpunkt der Rotationsachse mit der Ebene zurückgeführt wird. Dabei rotiert der Punkt $\mathbf{p}$ um den Punkt $\mathbf{p}_z$ zum Punkt $\mathbf{p}^{rot}$.

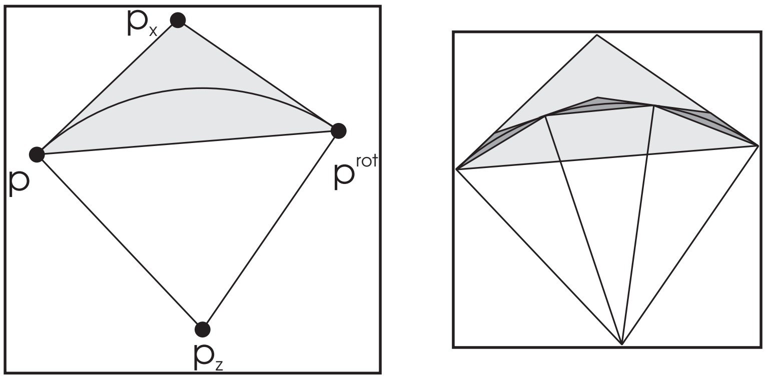

Abbildung 4.4: Rotation eines Punktes in der Ebene und einschließende Fläche

Betrachtet man nun die durch den Eckpunkt in der Rotationsebene beschriebene Bahn, dann entspricht diese, wie in Abbildung 4.4 (linkes Bild) zu sehen ist, einem Kreisbogen. Gesucht ist nun ein möglichst einfaches Polygon, das diesen Kreisbogen enthält. Dabei werden nur solche Kreisbögen betrachtet, die Rotationen aus dem Intervall $(0°, 180°)$ entsprechen. Rotationen mit einem Rotationswinkel $> 360°$ können durch eine Volldrehung des Polyeders ersetzt werden und dadurch kann jede Rotation in eine konstante Anzahl von Rotationen mit einem Rotationswinkel $< 180°$ zerlegt werden.

Zur Beschreibung des Polygons kann man das durch die Punkte $\mathbf{p}, \mathbf{p}^{rot}$ und $\mathbf{p}_\times$ gebildete Dreieck verwenden. Dabei ergibt sich der Punkt $\mathbf{p}_\times$ durch die Lösung des Gleichungssytems

$$\begin{aligned} \mathbf{p} + \alpha(\mathbf{p} - \mathbf{p}_z)^\perp &= \mathbf{p}^{rot} + \beta(\mathbf{p}^{rot} - \mathbf{p}_z)^\perp \\ \Leftrightarrow \quad \alpha(\mathbf{p} - \mathbf{p}_z)^\perp - \beta(\mathbf{p}^{rot} - \mathbf{p}_z)^\perp &= \mathbf{p}^{rot} - \mathbf{p} \end{aligned}$$

Durch die Beschränkung des Rotationswinkels auf Werte $< 180°$ verlaufen die beiden Geraden, deren Schnittpunkt berechnet wird, nicht parallel, und die Lösung des Gleichungssystems ist eindeutig bestimmt. Somit ergibt sich als Lösung:

$$\mathbf{p}_\times = \mathbf{p} + \frac{\det[\mathbf{p}^{rot} - \mathbf{p}, (\mathbf{p}^{rot} - \mathbf{p}_z)^\perp]}{\det[(\mathbf{p} - \mathbf{p}_z)^\perp, (\mathbf{p}^{rot} - \mathbf{p}_z)^\perp]}(\mathbf{p} - \mathbf{p}_z)^\perp$$

Diese Berechnung führt man für alle Ecken des Polyeders durch und erhält dadurch eine Menge von $|V_P| = O(|P|)$ Dreiecken. Für die Menge der Eckpunkte der Dreiecke wird die konvexe Hülle berechnet. Das dadurch berechnete Polyeder ist eine Approximation des Volumenkörpers, der bei der Rotation entsteht.

Je kleiner der betrachtete Rotationswinkel, desto genauer ist die Approximation des Kreisbogens durch das Dreieck. Daher ist es sinnvoll, für eine genauere Approximation des Volumenkörpers bei der Rotation den Rotationswinkel in mehrere kleinere Rotationswinkel zu unterteilen. Der Effekt der Unterteilung des Rotationswinkels in bezug auf die Genauigkeit der Approximation der Bahnkurve des Eckpunktes ist in Abbildung 4.4 (rechtes Bild) veranschaulicht.

Bei Rotationen rührt die Ungenauigkeit der Approximation des Volumenkörpers aus den beiden Problemen

- Ungenauigkeit bei der Approximation der Bahnkurven der Eckpunkte
- Ungenauigkeit bei der Approximation eines nichtkonvexen Volumenkörpers durch ein konvexes Polyeder

her. Indem die komplette Berechnung des Volumenpolyeders für kleinere Winkelintervalle durchgeführt wird, werden beide Ungenauigkeiten reduziert.

Lemma 4.3 :

Sei P ein konvexes Polyeder und $(\mathbf{r}, \theta)$ eine Rotation für P. Dann ist der durch die Rotation von P entstehende Volumenkörper $Vol_P^{(\mathbf{r},\theta)}$ vollständig in dem durch den Algorithmus berechneten Polyeder enthalten.

Beweis:
O.B.d.A. sei $\theta \in (0°, 180°)$, sonst ist durch den Algorithmus die Berechnung des Volumenkörpers in Teile mit Rotationen $< 180°$ zerlegt worden. Sei $Appr_P^{(\mathbf{r},\theta)}$ der durch den Algorithmus berechnete Polyeder und $\mathbf{p}$ ein beliebiger Punkt aus $Vol_P^{(\mathbf{r},\theta)}$.
z.z.: $\mathbf{p} \in Appr_P^{(\mathbf{r},\theta)}$

$\mathbf{p} \in Vol_P^{(\mathbf{r},\theta)} \Rightarrow \exists\ \alpha \in [0°, \theta]:\ \mathbf{p} \in P' = P(\mathbf{r}, \alpha)$
Sei $V_{P'}$ die Menge der Ecken von P'.
Dann läßt sich $\mathbf{p}$ darstellen als Konvexkombination der Ecken von P':

$$\mathbf{p} = \sum_{\mathbf{v} \in V_{P'}} \lambda_v \mathbf{v} \text{ mit } \sum_{\mathbf{v} \in V_{P'}} \lambda_v = 1,\ \lambda_v \geq 0\ \forall\ v \in V_{P'}$$

Nach Konstruktion der Dreiecke liegt jede Ecke $\mathbf{v} \in V_{P'}$ innerhalb der Dreiecksfläche, so daß sie sich als Konvexkombination der Eckpunkte des Dreiecks darstellen läßt, d.h. sei $V_{\triangle_v}$ die Eckenmenge des zu $\mathbf{v}$ gehörenden Dreiecks, so läßt sich $\mathbf{v}$ darstellen als

$$\mathbf{v} = \sum_{\mathbf{v}_\triangle \in V_{\triangle_v}} \mu_{v_\triangle} \mathbf{v}_\triangle \text{ mit } \sum_{\mathbf{v}_\triangle \in V_{\triangle_v}} \mu_{v_\triangle} = 1,\ \mu_{v_\triangle} \geq 0\ \forall\ v_\triangle \in V_{\triangle_v}$$

Somit läßt sich $\mathbf{p}$ darstellen als Linearkombination von Eckpunkten der Dreiecke durch

$$\mathbf{p} = \sum_{\mathbf{v} \in V_{P'}} \lambda_v \left(\sum_{\mathbf{v}_\triangle \in V_{\triangle_v}} \mu_{v_\triangle} \mathbf{v}_\triangle \right) = \sum_{\mathbf{v} \in V_{P'}} \sum_{\mathbf{v}_\triangle \in V_{\triangle_v}} \lambda_v \mu_{v_\triangle} \mathbf{v}_\triangle$$

Für die Summe der Linearfaktoren gilt:

$$\sum_{v \in V_{P'}} \sum_{v_\triangle \in V_{\triangle_v}} \lambda_v \mu_{v_\triangle} = \sum_{v \in V_{P'}} \lambda_v \underbrace{\left(\sum_{v_\triangle \in V_{\triangle_v}} \mu_{v_\triangle} \right)}_{=1} = \sum_{v \in V_{P'}} \lambda_v = 1\ \wedge\ \lambda_v \mu_{v_\triangle} \geq 0$$

Somit ist die obige Linearkombination eine Konvexkombination von Punkten, deren konvexe Hülle den Volumenkörper ergeben. Daher liegen alle Eckpunkte der Dreiecke in dem Volumenpolyeder, und wegen der Konvexität des berechneten Volumenpolyeders liegen alle Punkte, die sich als Konvexkombination von Punkten des Volumenpolyeders darstellen lassen, ebenfalls in dem Volumenpolyeder.

$\Rightarrow \mathbf{p} \in Appr_P^{(\mathbf{r},\theta)}$

■

Damit läßt sich die Vorgehensweise zusammenfassen zu folgendem Lemma:

Lemma 4.4 :

Gegeben sei ein Polyeder P und eine Rotation $(\mathbf{r}, \theta)$. Sei $Appr_P^{(\mathbf{r},\theta)}$ die Approximation des Körpers, der das während der Rotation von P überstrichene Volumen beschreibt, durch ein Polyeder.

1. *Ist P konvex, kann $Appr_P^{(\mathbf{r},\theta)}$ in Zeit $O(|P| \cdot \log |P|)$ berechnet werden und hat Komplexität $O(|P|)$.*
2. *Ist P nicht konvex und ist eine Zerlegung in k konvexe Teile bekannt, kann eine Menge von k konvexen Polyedern, deren Vereinigung $Appr_P^{(\mathbf{r},\theta)}$ bildet, in Zeit $O(k \cdot |P| \cdot \log |P|)$ berechnet werden, und die Vereinigung hat Komplexität $O(k^3 + k \cdot |P| \cdot \log^2 k)$.*
3. *Ist P nicht konvex, kann eine Menge von $O(|P|^2)$ konvexen Polyedern, deren Vereinigung $Appr_P^{(\mathbf{r},\theta)}$ bildet, in Zeit $O(|P|^2 \cdot \log |P|)$ berechnet werden, und die Vereinigung hat Komplexität $O(|P|^3)$.*

4.1.1.3 Volumenpolyeder bei allgemeinen Bewegungen

Für allgemeine Bewegungen stellt sich folgendes Problem:

gegeben:

P Polyeder
w Bewegung für P

gesucht:

$Vol_P^{(w)} = \{\mathbf{p} \in \mathbb{R}^3 \mid \exists\, t \in [\,0,1]: \ \mathbf{p} \in P(w(t))\}$

Bei allgemeinen Bewegungen enstehen i.a. Volumenkörper, die keine Polyeder sind, so daß diese analog zur Rotation approximiert werden durch Volumenpolyeder, die den tatsächlichen Volumenkörper enthalten.

Dabei benutzt man analog zur Rotation bei der Approximation der Volumenkörper nichtkonvexer Polyeder deren Zerlegung in konvexe Teile.

Um das Volumenpolyeder für P während der Bewegung w berechnen zu können, müssen für die Bewegung w die minimalen und maximalen Werte der Koordinaten eines Punktes während der Bewegung berechenbar sein.

Die Vorgehensweise für allgemeine Bewegungen ist eine Verallgemeinerung der Vorgehensweise bei Rotationen. Während sich bei einer Rotation eine Ecke des Polyeders in einer Ebene senkrecht zur Rotationsachse bewegt, gilt dies für allgemeine Bewegungen nicht mehr. D.h. es kann kein Polygon berechnet werden, das die Bahnkurve einer Ecke enthält, sondern ein Polyeder, das die Bahnkurve einer Ecke enthält.

Für jeden Eckpunkt $\mathbf{v}$ von P gilt:

$$\begin{pmatrix} x_{min} \\ y_{min} \\ z_{min} \end{pmatrix} \underset{\text{komponentenweise}}{\leq} \mathbf{v} \underset{\text{komponentenweise}}{\leq} \begin{pmatrix} x_{max} \\ y_{max} \\ z_{max} \end{pmatrix}$$

D.h. die Bahnkurve von $\mathbf{v}$ ist vollständig in dem Quader mit den Eckpunkten

$$\begin{pmatrix} x_{min} \\ y_{min} \\ z_{min} \end{pmatrix}, \begin{pmatrix} x_{max} \\ y_{min} \\ z_{min} \end{pmatrix}, \begin{pmatrix} x_{max} \\ y_{max} \\ z_{min} \end{pmatrix}, \begin{pmatrix} x_{min} \\ y_{max} \\ z_{min} \end{pmatrix}, \begin{pmatrix} x_{min} \\ y_{min} \\ z_{max} \end{pmatrix}, \begin{pmatrix} x_{max} \\ y_{min} \\ z_{max} \end{pmatrix}, \begin{pmatrix} x_{max} \\ y_{max} \\ z_{max} \end{pmatrix}, \begin{pmatrix} x_{min} \\ y_{max} \\ z_{max} \end{pmatrix}$$

enthalten. Berechnet man nun wie bei den Rotationen als Volumenpolyeder die konvexe Hülle der Eckpunkte aller dieser Quader, so erhält man analog zur Rotation ein Volumenpolyeder, das den tatsächlichen Volumenkörper enthält.

Der Beweis für die Eigenschaft des erhaltenen Volumenpolyeders verläuft analog zu dem bei Rotationen. Aufgrund der Vorgehensweise ergeben sich für allgemeine Bewegungen dieselben Laufzeiten wie für Rotationen. Diese lassen sich in folgendem Lemma zusammenfassen:

Lemma 4.5 :

Gegeben sei ein Polyeder P und eine Bewegung w. Sei $Appr_P^w$ das Polyeder, welches das von P während der Bewegung w überstrichene Volumen approximiert.

1. *Ist P konvex, kann $Appr_P^w$ in Zeit $O(|P| \cdot \log|P|)$ berechnet werden und hat Komplexität $O(|P|)$.*
2. *Ist P nicht konvex und ist eine Zerlegung von P in k konvexe Teile bekannt, so kann eine Menge von k konvexen Polyedern, deren Vereinigung $Appr_P^w$ bildet, in Zeit $O(k \cdot |P| \cdot \log|P|)$ berechnet werden und die Vereinigung hat Komplexität $O(k^3 + k \cdot |P| \cdot \log^2 k)$.*
3. *Ist P nicht konvex, so kann eine Menge von $O(|P|^2)$ Polyedern, deren Vereinigung $Appr_P^w$ bildet, in Zeit $O(|P|^2 \cdot \log|P|)$ berechnet werden und die Vereinigung hat Komplexität $O(|P|^3)$.*

Für jede Bewegung w kann somit eine Menge von Polyedern berechnet werden, deren Vereinigung im Falle von rein translatorischen Bewegungen das überstrichene Volumen eines Polyeders P exakt beschreibt bzw. im Falle anderer Bewegungen approximiert. Mit Hilfe dieser Polyeder kann nun die Menge der Ausweichpositionen für die störenden, bewegbaren Hindernisse in der Szene berechnet werden.

Mit Hilfe des Bewegungsplanungsalgorithmus für das bewegbare Hindernis $h_{bewegbar}$, der auf dem Konfigurationsraumansatz beruht, können nun die Konfigurationsraumhindernisse berechnet werden, die sich durch eine Szene mit den berechneten Volumenpolyedern als Hindernissen ergeben. Die Berechnung der Konfigurationsraumhindernisse der als fest angesehenen Hindernisse erfolgt analog. Dadurch erhält man den Freiraum für das bewegbare Hindernis bzgl. der als fest angesehenen Hindernisse und der Schnitt des Freiraums mit dem Freiraum im Konfigurationsraum von $h_{bewegbar}$ mit den berechneten Volumenpolyedern ergibt die Menge der freien Endpunkte der Ausweichpositionen.

Die Berechnung der möglichen Endpositionen einer Ausweichbewegung für ein Hindernis wird nun noch an einem Beispiel veranschaulicht.

4.1.2 Ein Beispiel

In Anlehnung an den Bewegungsplanungsalgorithmus, der im Rahmen dieser Arbeit innerhalb des IGOR-Systems realisiert wurde, wird ein Beispiel betrachtet, in dem für das bewegte Objekt eine translatorische Bewegung berechnet wurde und für eines der bewegbaren Hindernisse die Endpositionen einer translatorischen Ausweichbewegung in einer Ebene berechnet wird.

Als Beispiel soll die Problemstellung in Abbildung 4.9 auf Seite 94 betrachtet werden, für das in *Anhang B* zusammenfassend auch eine vollständige Lösung mit Hilfe der gesamten Heuristik beschrieben wird. An dieser Stelle wird die Betrachtung auf eines der beiden bewegbaren Hindernisse beschränkt.

Für das bewegte Objekt P wurde als Bewegung die in Abbildung 4.5 veranschaulichte translatorische Bewegung berechnet, die zu dem daneben dargestellten Volumenpolyeder Vol_P führt.

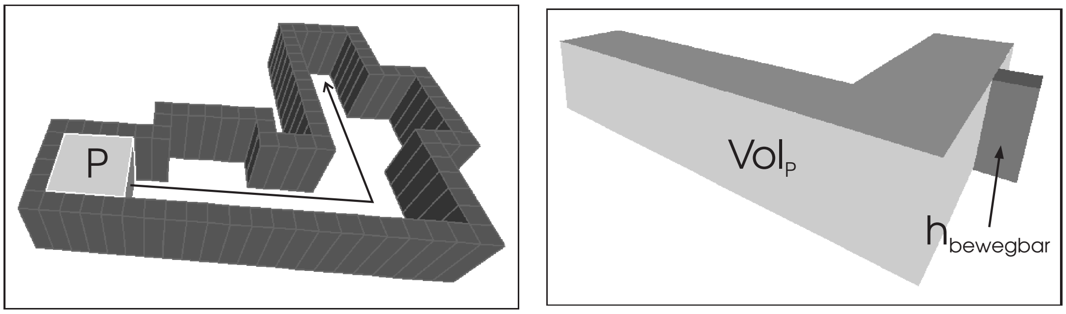

Abbildung 4.5: Berechnete translatorische Bewegung und zugehöriges Volumenpolyeder mit bewegbaren Hindernis

Daraus ergibt sich als Konfigurationsraumhindernis für das bewegbare Hindernis bzgl. des Volumenpolyeders das aus Abbildung 4.6 *(Bild rechts oben)* ersichtliche Polygon CO_{Vol}. Aus der Betrachtung der als fest angesehenen Hindernisse ergibt sich die in Abbildung 4.6 *(Bild links oben)* ersichtliche polygponale Region CO_h. Der Schnitt der beiden Freiräume ergibt die in Abbildung 4.6 *(Bild unten)* dargestellten Endpositionen der Ausweichbewegung.

4.2 Berechnung der Ausweichbewegungen

Nach der intuitiven Idee der Strategie ergeben sich i.a. kurze Ausweichbewegungen, zu deren Beschreibung wenige Hindernisse in der Szene beitragen. Deshalb ist zu deren Berechnung die Betrachtung lokal eingeschränkter Szenarien sinnvoll, um den Berechnungsaufwand zu verringern.

Dazu kann z.B. die Beschränkung der Szene auf einen Quader oder eine Kugel um das bewegbare Hindernis benutzt werden. Diese Umgebung wird solange vergrößert, bis eine Ausweichbewegung darin berechnet werden oder entschieden werden kann, daß keine solche existiert.

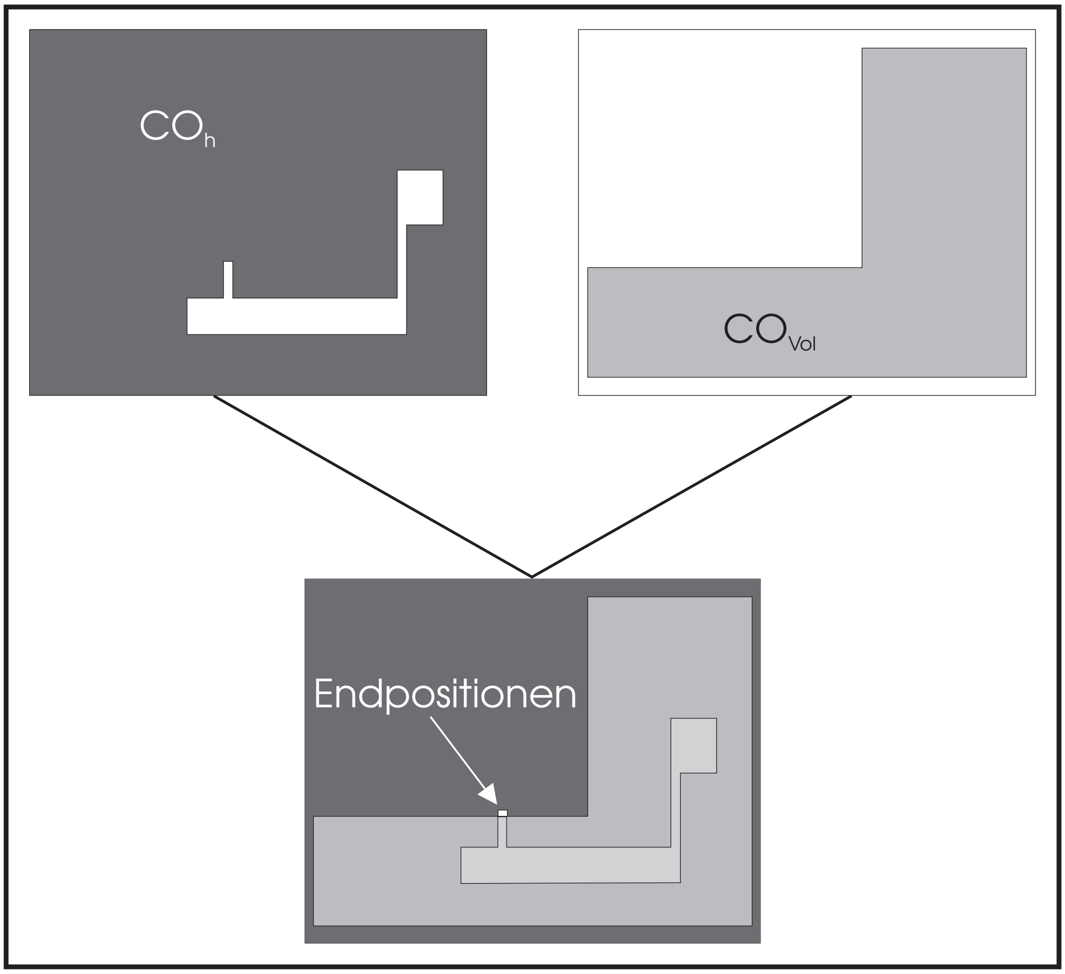

Abbildung 4.6: Resultierendes Konfigurationsraumhindernis und mögliche Endpositionen der Ausweichbewegung

Sind mehrere Ausweichbewegungen zu berechnen, so stellt sich die Frage der Koordination der verschiedenen Ausweichbewegungen. Diese kann

- zusammen mit der Bewegungsberechnung,
- in einem separaten Koordinationsschritt

erfolgen.

Die Ausweichbewegungen der einzelnen bewegbaren Hindernisse werden vor der Bewegung des bewegten Objektes durchgeführt, so daß das bewegte Objekt in seiner Startposition bei der Berechnung der Ausweichbewegungen als Hindernis berücksichtigt werden muß.

4.2.1 Gleichzeitige Berechnung und Koordination der Ausweichbewegungen

Eine einfache Möglichkeit, die Berechnung und die Koordination der Ausweichbewegungen simultan durchzuführen, besteht darin, für die einzelnen bewegbaren Hindernisse a priori Prioritäten zu vergeben, so daß eine Reihenfolge zur Berechnung der Ausweichbewegungen für die einzelnen bewegbaren Hindernisse gegeben ist.

Entsprechend dieser Reihenfolge werden die Bewegungen für die bewegbaren Hindernisse berechnet und dabei bewegbare Hindernisse, für die die Ausweichbewegungen noch berechnet werden müssen, in ihrer Ausgangsposition und bewegbare Hindernisse, für die die Ausweichbewegungen bereits berechnet sind, in ihrer Endposition als feste Hindernisse in die Szene eingefügt.

Durch diese Vorgehensweise ergibt sich eine Folge von Bewegungen für die einzelnen störenden Hindernisse, die – sukzessive abgearbeitet – kollisionsfrei ist. Dadurch können Ausweichbewegungen mehrerer Objekte berechnet werden, bei denen keine zeitliche Koordination zwischen den Bewegungen nötig ist und keine Konkurrenz bei der Wahl der Endpositionen vorliegt. Kommt es zu einer Konkurrenz zwischen mehreren Objekten bei der Wahl der Endposition der Bewegung, so kann durch die Priorität der Objekte eine Lösung verhindert werden. Ausweichbewegungen, die eine zeitliche Koordination von Bewegungen erfordern, wie z.B. in Abbildung 4.7, können mit dieser Strategie nicht berechnet werden.

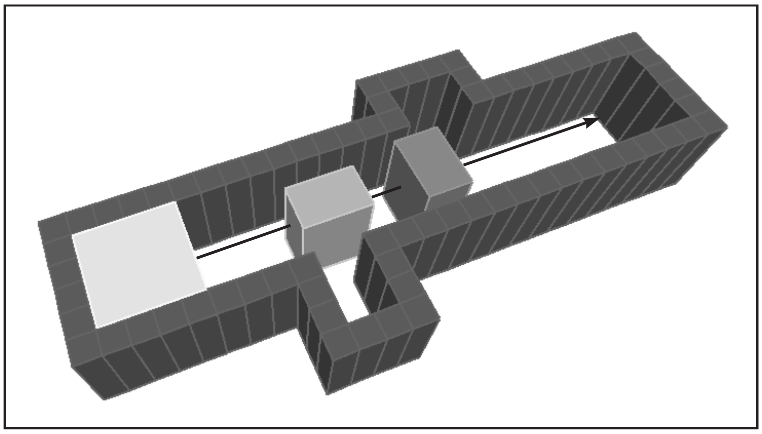

Abbildung 4.7: Verhinderung einer Lösung durch sukzessive Planung

Für den Fall zweier translatorischer Freiheitsgrade bietet der Algorithmus von Erdman und Lozano-Perez [ELP87] die Möglichkeit, bei der Berechnung von Ausweichbewegungen zeitliche und örtliche Koordination von Bewegungen zur Problemlösung zu nutzen. Dadurch ist der Algorithmus z.B. in der Lage, das Problem in Abbildung 4.7 zu lösen. Es bleibt jedoch das Problem, daß durch die Prioritäten der bewegten Objekte Probleme bei der Wahl der Endpositionen zwischen verschiedenen Objekten nicht immer gelöst werden können.

Bei dem Beispiel in Abbildung 4.8 findet der Algorithmus keine Lösung, falls der kleine Würfel die höhere Priorität hat und als Endposition die große Nische wählt.

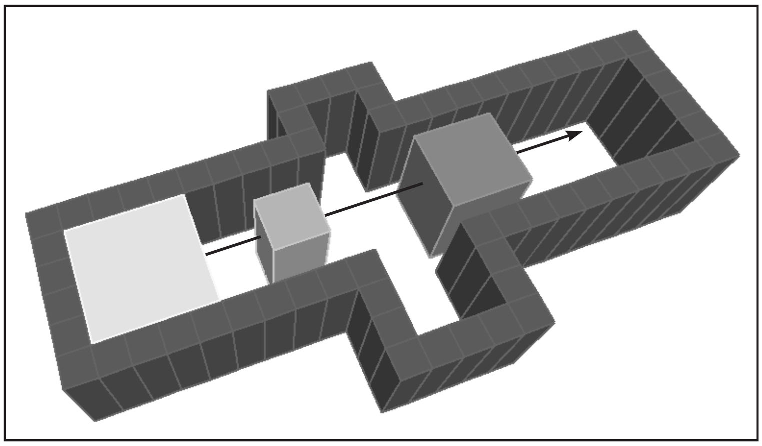

Abbildung 4.8: Konkurrenz bei der Wahl der Endpositionen

Zusätzlich ergibt sich für den Algorithmus das Problem, daß er nicht in der Lage ist, Probleme zu lösen, bei denen zwei oder mehrere Objekte nacheinander jeweils in Ausweichpositionen bewegt werden müssen, um dadurch die Koordination der Bewegungen der Objekte in ihre jeweilige Endposition zu ermöglichen. Dadurch können Probleme wie in Abbildung 4.9 nicht gelöst werden. Dazu ist eine Erweiterung der Strategie notwendig, die Bewegungsberechnung und -koordination trennt.

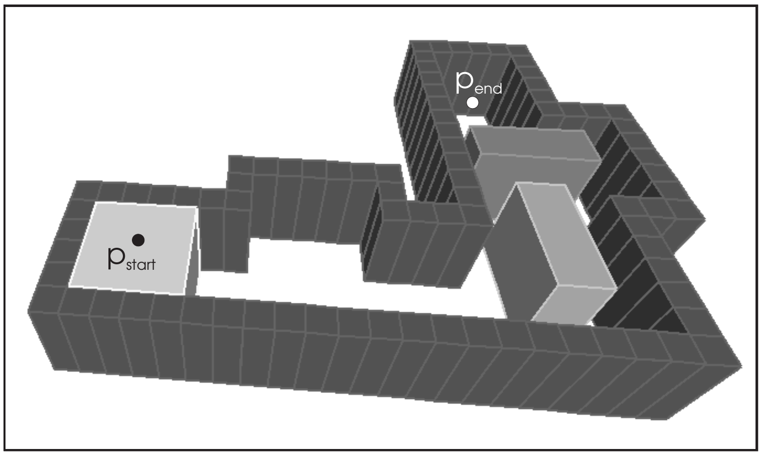

Abbildung 4.9: Lösung nur mit Hilfe mehrfacher Bewegungen

4.2.2 Trennung von Bewegungsberechnung und -koordination

Durch die simultane Berechnung und Koordination von Ausweichbewegungen mit Hilfe der oben angegebenen Verfahren können wegen der a priori Vergabe von Prioritäten für die Hindernisse Konflikte bei der Wahl der Endpositionen der Ausweichbewegungen nicht sicher gelöst werden. Dadurch sind beide Vorgehensweisen nicht in der Lage, das Problem in Abbildung 4.8 immer zu lösen. Dazu ist eine Strategie nötig, die Konflikte bei der Wahl der Endpositionen zu lösen vermag.

Die Vorgehensweise von Erdmann und Lozano-Perez zeigt die Schwäche, daß sie neben der Beschränkung auf zwei translatorische Freiheitsgrade, Wege für Objekte immer vollständig plant, so daß ein Problem wie in Abbildung 4.9, bei dem mehrere Objekte nacheinander jeweils zunächst in eine Ausweichposition und dann erst in die eigentliche Zielposition bewegt werden müssen, nicht gelöst werden kann.

Viele Bewegungsplanungsalgorithmen berechnen zur Lösung des Bewegungsplanungsproblems einen Graphen (z.B. einen Sichtbarkeitsgraphen oder ein Voronoidiagramm), in dem ein Weg von der Start- zur Endposition berechnet wird. D.h. es wird ein Weg aus dem Graphen ausgewählt und die Kanten, die nicht auf dem Weg liegen, werden bei der Lösung vernachlässigt. Diese bieten diskrete örtliche Ausweichmöglichkeiten bei der Koordination von Bewegungen.

Daher kann man davon ausgehen, daß für jedes der bewegbaren Hindernisse eine Straßenkarte von kollisionsfreien Bewegungsmöglichkeiten berechnet wird, die im Extremfall nur aus einem einzigen Pfad besteht.

Bemerkung 4.6 :

> *Wird zur Berechnung des Weges die heuristisch inkrementelle Vorgehensweise verwendet, so kann in einem berechneten Graphen (z.B. Sichtbarkeitsgraph oder Voronoidiagramm) nicht für alle Kanten garantiert werden, daß die zugehörige Bewegung kollisionsfrei ist. In diesem Fall müssen neben dem berechneten Pfad alle anderen Kanten des Graphen auf Kollisionen getestet und der Graph durch Streichen von nicht kollisionsfreien Kanten modifiziert werden.*

4.2.2.1 Unabhängige Berechnung der Ausweichbewegungen

Für jedes der störenden, bewegbaren Hindernisse wird unabhängig von den anderen störenden Hindernissen eine Ausweichbewegung berechnet. Dabei werden als feste Hindernisse die ursprünglich in der Szene als feste Hindernisse vorhandenen Objekte, die bewegbaren Hindernisse, die die berechnete Bewegung des bewegten Objektes nicht stören, sowie das bewegte Objekt in seiner Startposition berücksichtigt. Dadurch entsteht für jedes der bewegbaren Hindernisse eine Straßenkarte von kollisionsfreien Bewegungen.

Bemerkung 4.7 :

> *Alle nichtstörenden, bewegbaren Hindernisse in der Szene werden als fest angenommen, so daß bei der Koordination der einzelnen Ausweichbewegungen zusätzliche Nebenbedingungen vermieden werden, die die während der Ausweichbewegungen beseitigten bewegbaren Hindernisse betreffen.*

4.3 Koordination der Ausweichbewegungen

Für jedes der störenden bewegbaren Hindernisse ist eine Straßenkarte von Bewegungsmöglichkeiten in Form eines Graphen berechnet. Gesucht ist ein Weg in einer Gesamtstraßenkarte, die alle Bewegungen koordiniert enthält. Formal stellt sich somit folgendes Problem:

gegeben:

$H^{col}_{bewegbar}$ Menge der störenden, bewegbaren Hindernisse

$$S_{H^{col}_{bewegbar}} = \left\{ (h^{col}_{bewegbar}, G_{h^{col}_{bewegbar}}) \;\middle|\; \begin{array}{l} G_{h^{col}_{bewegbar}} \text{ kollisionsfreie Straßenkarte für } h^{col}_{bewegbar} \\ h^{col}_{bewegbar} \in H^{col}_{bewegbar} \end{array} \right\}$$

gesucht:

$$w_{koordiniert} = \text{Weg in } G_{koordiniert}(S_{H^{col}_{bewegbar}})$$

Als Koordinationsgraph $G_{koordiniert}(S_{H^{col}_{bewegbar}})$ wird der kartesische Produktgraph der einzelnen Graphen $G_{h^{col}_{bewegbar}}$ mit $h^{col}_{bewegbar} \in H^{col}_{bewegbar}$ berechnet.

Definition 4.8 (Koordinationsgraph)

Sei H eine Menge von Objekten und für jedes Objekt h eine Straßenkarte G_h gegeben, die zu einem Tupel $S_H = (G_{h_1}, \ldots, G_{h_{|H|}})$ von Straßenkarten zusammengefaßt wird. Der Graph $G_{koordiniert}(S_H)$ mit

$$\begin{aligned} V_{G_{koordiniert}} &= \{(v^1, \ldots, v^{|H|}) \mid v^i \in V_{G_{h_i}},\ i = 1, \ldots, |H|\} \\ E_{G_{koordiniert}} &= \{\{v_i, v_j\} \mid \exists \text{ Koordination der Bewegungen } v_i^1 \to v_j^1, \ldots, v_i^{|H|} \to v_j^{|H|}\} \end{aligned}$$

heißt **Koordinationsgraph** *von S_H.*

Zur Berechnung des Koordinationsgraphen müssen im wesentlichen die Kanten des Graphen berechnet werden. Dazu werden simultane Einzelbewegungen der Objekte betrachtet. Ist eine Koordination dieser möglich, enthält der Koordinationsgraph die entsprechende Kante.

Für die Koordination von Einzelbewegungen sind verschiedene Vorgehensweisen denkbar:

1. Vernachlässigung des Zeitaspektes
2. Geschwindigkeitsanpassung mehrerer Bewegungen

4.3.1 Vernachlässigung des Zeitaspektes

Wird bei der Koordination von Einzelbewegungen der Zeitaspekt vernachlässigt, entspricht dies der Projektion des 4-dimensionalen Objekt-Zeit-Raumes auf den 3-dimensionalen Objektraum, in dem die Bewegungen als überstrichene Volumenkörper der einzelnen Objekte repräsentiert werden. Ist der Schnitt der Volumenkörper der Objekte leer, ist eine simultane kollisionsfreie Bewegung aller Objekte bei jeder zeitlichen Verteilung der Bewegungen möglich.

Dies ist eine einfache Möglichkeit zur Berechnung der Kanten des Koordinationsgraphen. Dabei werden Geschwindigkeiten von Bewegungen der einzelnen Körper nicht in Betracht gezogen.

4.3.2 Geschwindigkeitsanpassung der Objekte

Kant und Zucker haben in [KZ86] einen Algorithmus beschrieben, der das Problem der Berechnung einer Bewegung für ein Objekt bei bewegten Hindernissen unterteilt in die beiden Teilprobleme

- Berechnung eines Weges für das zu bewegende Objekt ohne Berücksichtigung der bewegten Hindernisse
- Koordination der Bewegungen durch geschwindigkeitsmäßige Anpassung

Davon kann an dieser Stelle der zweite Teil des Algorithmus benutzt werden. Kant und Zucker zeigen, daß für die Bewegungen ein 2-dimensionaler Konfigurationsraum berechnet werden kann, in dem das Problem der geschwindigkeitsmäßigen Anpassung auf das Problem des Findens eines zeitmonotonen Weges in einem Weglängen-Zeit-Konfigurationsraum zurückgeführt werden kann.

Dadurch kann für die Einzelbewegungen zweier bewegbarer Hindernisse, für die eine zeitliche Koordination nötig ist, ein Geschwindigkeitsprofil für eines der beiden Hindernisse berechnet werden, wobei das andere ein konstantes Geschwindigkeitsprofil erhält.

Jedoch kann auch für die Berechnung unter *Vernachlässigung des Zeitaspektes* eine diskrete Koordination durch zeitliche Anpassung dadurch erreicht werden, daß die einzelnen Kanten der individuellen Straßenkarten in kleinere Teile zerlegt werden.

Nachdem Möglichkeiten zur Berechnung des Koordinationsgraphen beschrieben wurden, verbleibt als Problem noch die Wegesuche im Koordinationsgraphen.

4.3.3 Wegesuche im Koordinationsgraphen

Eine Möglichkeit zur Wegesuche besteht darin, den Koordinationsgraphen in einem ersten Schritt vollständig zu berechnen und dann in diesem Graphen einen Weg vom Knoten $v_{start} = (v^1_{start}, \ldots, v^{|H|}_{start})$ zu einem Knoten $v_{end} = (v^1_{end}, \ldots, v^{|H|}_{end})$ mit $v^i_{end} \in P^i_{end}$ für $i = 1, \ldots, |H|$ zu suchen. Dabei kann als Optimalitätskriterium die Summe der Längen der Wege der einzelnen Objekte benutzt werden. Dadurch ergeben sich nichtnegative Kantengewichte und Dijkstras Algorithmus zur Berechnung kürzester Wege kann zur Berechnung eines optimalen Weges in dem Graphen benutzt werden.

Da dieser Koordinationsgraph als kartesischer Produktgraph anderer Graphen groß werden kann und nur ein Weg von v_{start} nach v_{end} in diesem Graph interessant ist, kann die Berechnung des gesamten Graphen heuristisch mit Hilfe des A*-Algorithmus (vgl. [Pe84]) vermieden werden.

Damit der A*-Algorithmus eine optimale Lösung findet, muß für das Optimalitätskriterium eine Bewertungsfunktion für die Knoten des Koordinationsgraphen gefunden werden, die die Länge des Restweges unterschätzt. Betrachtet man als Optimialitätskriterium für den Weg im Koordinationsgraphen die Minimierung der Summe aller Längen der Ausweichbewegungen der bewegbaren Hindernisse, so ergibt sich als minimale Länge des Weges im Koordinationsgraphen die Summe der kürzesten Wege der bewegbaren Hindernisse in ihren individuellen Straßenkarten. Man unterschätzt die Länge des Restweges im Koordinationsgraphen sicher dadurch, daß man für jedes bewegbare Hindernis von seinem individuell kürzesten Weg die Länge des – zum aktuellen Zeitpunkt – bereits zurückgelegten Weges subtrahiert und alle sich dadurch ergebenden nichtnegativen Elemente addiert. Als Bewertungsfunktion für die Knoten des kartesischen Produktgraphen ergibt sich somit

$$h(v) = \sum_{i=1}^{|H|} \max\{0, |w^i_{min}| - |v^i_{start} \rightarrow v^i|\}$$

wobei w^i_{min} die Länge des kürzesten Weges in der Straßenkarte des i-ten Objektes und $|v^i_{start} \rightarrow v^i|$ die Länge des Weges ist, den das i-te Objekte von v^i_{start} nach v^i zurücklegt.

Können alle bewegbaren Hindernisse ihre kürzesten Bewegungen durchführen, findet der A*-Algorithmus ohne Umwege den Weg im Koordinationsgraphen. Müssen alle Objekte einen längeren Weg durchführen als ihren individuell kürzesten, führt der A*-Algorithmus zu einer gewichteten Breitensuche, wenn alle bewegbaren Hindernisse mindestens ihre individuell kürzeste Wegstrecke zurückgelegt haben.

Die Anwendung des A*-Algorithmus ist insbesondere deswegen interessant, weil er versucht, die Größe des während der Wegesuche expandierten Graphen so klein wie möglich zu halten. Da die Berechnung der Kanten des Koordinationsgraphen teuer ist, wird durch diese heuristische Vorgehensweise versucht, die Laufzeit der Bewegungskoordination zu verringern.

4.3.4 Zusammenfassung der Bewegungskoordination

Für die Koordination von Ausweichbewegungen bieten sich verschiedene Möglichkeiten an, die mehr oder weniger mächtig sind. Für Probleme, bei denen die Hindernisse nicht dicht zusammenstehen und daher Koordinationsprobleme zwischen den einzelnen störenden Hindernissen weniger zu erwarten sind, bietet die sukzessive Planung von Ausweichbewegungen für die bewegbaren Hindernisse eine einfache Möglichkeit, das Koordinationsproblem zu lösen. Für den speziellen Fall von zwei translatorischen Freiheitsgraden ergibt sich durch den Algorithmus von Erdmann und Lozano-Perez [ELP87] die Möglichkeit, zeitliche und örtliche Koordination der Ausweichbewegungen heuristisch durchzuführen.

Ist eine komplexere Koordination von Ausweichbewegungen notwendig, bietet sich die Betrachtung einer Straßenkarte für jedes Objekt an, die diskrete örtliche Ausweichmöglichkeiten bietet. Durch die Bildung eines Koordinationsgraphen aus den einzelnen individuellen Straßenkarten ergibt sich eine Möglichkeit, die diskreten Ausweichmöglichkeiten mit der zeitlichen Koordination der Bewegungen zu verbinden. Dabei hängt die Gewichtung von zeitlicher und örtlicher Koordination der Bewegung von der Art und Feinheit der gewählten Straßenkarte ab. Grundsätzlich möglich ist jede Art der Straßenkarte. Zur Betonung der diskreten Ausweichmöglichkeiten bietet sich jedoch bei translatorischen Bewegungen ein Gitter über dem Konfigurationsraum an, in dem jeweils zwei benachbarte Gitterpunkte miteinander verbunden werden, die eine direkte Bewegung zulassen (vgl. Abbildung 4.10).

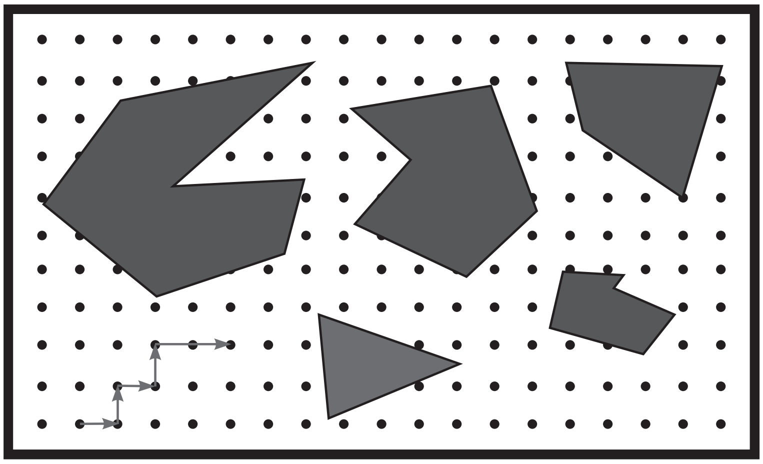

Abbildung 4.10: Gitter im Konfigurationsraum

Je größer die Straßenkarten der einzelnen Objekte sind, desto größer wird der zu berechnende Koordinationsgraph. Dabei bietet die Berechnung eines Weges im Koordinationsgraph mit Hilfe des A*-Algorithmus die Möglichkeit, heuristisch die Größe des während der Wegesuche zu expandierenden Graphen zu verringern. Wird der Koordinationsgraph zunächst vollständig berechnet, kann darin ein optimaler Weg mit Hilfe bekannter Algorithmen wie Dijkstras Algorithmus berechnet werden.

4.4 Zusammenfassung

In diesem Kapitel wurde ein Algorithmus vorgestellt, der die klassische Bewegungsplanung dahingehend erweitert, daß die Hindernisobjekte in einer Szene in zwei Kategorien – feste Hindernisse und bewegbare Hindernisse – eingeteilt werden, wobei die bewegbaren Hindernisse zur Lösung des Bewegungsplanungsproblems für ein bewegtes Objekt von ihrer Ausgangsposition wegbewegt werden können.

Formell entspricht diese Problemstellung einem Bewegungsplanungsproblem mit mehreren bewegten Objekten, wobei nur für eines der Objekte – das bewegte Objekt – eine Zielposition angegeben ist. Es handelt sich um eine Instanz des allgemeinen Bewegungsplanungsproblems und kann daher exakt mit Hilfe des Algorithmus von Sharir und Schwartz [SS83b] gelöst werden. Dieses Verfahren ist jedoch nicht praktikabel.

Gesucht war ein praktikabler heuristischer Algorithmus zur Lösung dieses Bewegungsplanungsproblems. Ausgangspunkt war dabei die Beobachtung der Vorgehensweise eines menschlichen Planers bei der Lösung eines solchen Bewegungsplanungsproblems:

> Ein menschlicher Planer klassifiziert die Hindernisse in der Szene implizit in die beiden Kategorien *bewegbare Hindernisse* und *feste Hindernisse* und vernachlässigt bei der Wegeplanung für sein bewegtes Objekt die bewegbaren Hindernisse. Stören nach der Planung eines kollisionsfreien Weges für das bewegte

Objekt bzgl. der festen Hindernisse bewegbare Hindernisse diesen Weg, werden diese zunächst aus dem Weg gebracht und dann wird die zuvor geplante Bewegung für das bewegte Objekt durchgeführt.

Dabei fällt insbesondere auf, daß in der Regel nur eine kleine Anzahl der als beweglich klassifizierten Hindernisse für eine Problemlösung bewegt werden muß und die Betrachtung auf diese eingeschränkt wird, sobald ein Weg für das bewegte Objekt geplant ist.

Diese Beobachtung ergibt den Ansatzpunkt für einen heuristischen Algorithmus, dessen Teilalgorithmen in diesem Kapitel beschrieben wurden.

Dabei ergeben sich grundsätzlich zwei Verwendungstrategien für die Heuristik:

1. **Aufsatz auf die klassische Bewegungsplanung**

   In einem ersten Schritt wird versucht, das Bewegungsplanungsproblem mit Hilfe eines klassischen Bewegungsplanungsalgorithmus und ausschließlich festen Hindernissen zu lösen. Mißlingt diese Vorgehensweise, wird die Klassifizierung der Hindernisse in *feste Hindernisse* und *bewegbare Hindernisse* vorgenommen und die heuristische Bewegungsplanung bei bewegbaren Hindernissen durchgeführt.

   Aufgrund der Vorgehensweise ist ersichtlich, daß die Heuristik jedes Bewegungsplanungsproblem löst, das ein klassischer Algorithmus löst, und daß durch die Möglichkeiten der Bewegung von Hindernissen die Lösungsmöglichkeiten des klassischen Bewegungsplanungsalgorithmus erweitert werden.

   Die Heuristik kann jedoch nicht nur als Aufbau auf einen klassischen Bewegungsplanungsalgorithmus benutzt werden, sondern erweitert insbesondere auch bei direkter Anwendung die Lösungsmöglichkeiten klassischer Bewegungsplanungsalgorithmen.

2. **Direkte Anwendung der Heuristik**

   Die Heuristik direkt angesetzt auf ein Bewegungsplanungsproblem erweitert ebenfalls die Lösungsmöglichkeiten der klassischen Bewegungsplanung, wenn sie um eine Problemmodifikationsstrategie im Mißerfolgsfall erweitert wird.

   Findet die Heuristik keine Lösung, so liegt dies an einem der folgenden Probleme:

   (1) Der Bewegungsplanungsalgorithmus für das bewegte Objekt findet keine Bewegung bei Berücksichtigung nur der festen Hindernisse.

   (2) Für ein oder mehrere störende, bewegbare Hindernisse kann keine Ausweichbewegung berechnet werden.

   (3) Eine Koordination der Ausweichbewegungen der bewegbaren, störenden Hindernisse ist nicht möglich.

   Im ersten Fall hat das klassische Bewegungsplanungsproblem ebenfalls keine Lösung. Im zweiten Fall werden diese bewegbaren Hindernisse als fest klassifiziert und die Heuristik erneut gestartet. Im dritten Fall wird aus der Menge der störenden, bewegbaren Hindernisse randomisiert eines als fest klassifiziert und die Heuristik erneut gestartet.

Im zweiten und dritten Fall ergibt sich durch die Modifikation der Szene ein Bewegungsplanungsproblem, bei dem die Anzahl der bewegbaren Hindernisse kleiner ist als bei dem nichtmodifizierten. Da die Anzahl der bewegbaren Hindernisse endlich ist, ergibt sich nach höchstens Anzahl der bewegbaren Hindernisse vielen Iterationen das klassische Bewegungsplanungsproblem, das dann ausschließlich mit Hilfe des Bewegungsplanungsalgorithmus für das bewegte Objekt bearbeitet wird.

Dadurch ist auch für diese Vorgehensweise folgender Satz gezeigt:

Satz 4.9:
Sei BPA ein klassischer Bewegungsplanungsalgorithmus, BPP ein klassisches Bewegungsplanungsproblem und $BPP_{bewegbar}$ eine Modifikation von BPP, bei der Hindernisse als bewegbar klassifiziert wurden. Dann gilt:

(1) Findet BPA angesetzt auf BPP eine Lösung, so findet die heuristische Strategie angesetzt auf $BPP_{bewegbar}$ ebenfalls eine Lösung, d.h. die Menge L_{BPA} der von BPA lösbaren Bewegungsplanungsprobleme ist eine Teilmenge der von BPA mit der heuristischen Strategie lösbaren Bewegungsplanungsprobleme $L_{BPA}^{bewegbar}$.

$$L_{BPA} \subseteq L_{BPA}^{bewegbar}$$

(2) Findet BPA angesetzt auf BPP keine Lösung, so kann die heuristische Strategie angesetzt auf $BPP_{bewegbar}$ eine Lösung finden, d.h. sei $l_{BPA}(BPP)$ eine Lösung von BPP berechnet durch BPA und $l_{BPA}(BPP_{bewegbar})$ eine Lösung von $BPP_{bewegbar}$ berechnet durch die heuristische Strategie, dann gilt:

$$\exists\ BPA, BPP, BPP_{bewegbar}:$$
$$l_{BPA}(BPP) \text{ existiert nicht} \quad \wedge \quad l_{BPA}^{bewegbar}(BPP_{bewegbar}) \text{ existiert}$$

Bemerkung 4.10:

(a) Die erste Aussage des Satzes zeigt, daß die heuristische Bewegungsplanungsstrategie auf keinen Fall die Lösungsmöglichkeiten eines klassischen Bewegungsplanungsalgorithmus einschränkt.

(b) Die zweite Aussage zeigt, daß die heuristische Bewegungsplanungsstrategie tatsächlich die Lösungsmöglichkeiten eines klassischen Bewegungsplanungsalgorithmus erweitern kann. Zwar kann man theoretisch nur zeigen, daß es Beispiele gibt, die nur von der heuristischen Bewegungsplanungsstrategie und nicht von dem klassischen Algorithmus gelöst werden können, in der Praxis stellt dieses Konzept eine deutliche Erweiterung der Lösungsmöglichkeiten klassischer Bewegungsplanungsalgorithmen dar, wie auch anhand der Beispiel in Anhang B *zu sehen ist.*

4.4.1 Interaktion mit der heuristisch inkrementellen Vorgehensweise

Zusätzlich kann die in diesem Kapitel beschriebene Heuristik zusammen mit der in *Kapitel 2* beschriebenen heuristisch inkrementellen Vorgehensweise verwendet werden. Diese ist insbesondere sinnvoll für die Berechnung der Bewegung für das bewegte Objekt unter Berücksichtigung nur der festen Hindernisse.

Es ergibt sich dabei folgender Gesamtalgorithmus:

P	bewegtes Objekt
BPA	Bewegungsplanungsalgorithmus zur Berechnung der Bewegung für P
$\mathbf{p}_{start}$	Startposition von P
$\mathbf{p}_{end}$	Endposition von P
w	Weg für das bewegte Objekt
H_b	Menge bewegbarer Hindernisse
h_b	bewegbares Hindernis in H_b
H_b^{col}	Teilmenge der bewegbaren Hindernisse, die w für P behindern
BPA_{h_b}	Bewegungsplanungsalgorithmus für $h_b \in H_b$
M_{w_b}	Menge von Ausweichbewegungen für die Hindernisse aus H_b^{col}
H_f	Menge fester Hindernisse
$success$	Boolesche Variable zur Anzeige des Erfolges des Algorithmus

$sucess \leftarrow TRUE$
$M_{w_b} \leftarrow \emptyset$
if $(\neg(w \leftarrow HeuristischInkrementelleBewegungsplanung(BPA, P, H_f, \mathbf{p}_{start}, \mathbf{p}_{end})))$ **then**
    $success \leftarrow FALSE$
    end
$H_b^{col} \leftarrow Kollision(P, H_b, w)$
if $(H_b^{col} \neq \emptyset)$ **then**
    forall $h_b \in H_b^{col}$ **do**
        if $(\neg(w_b \leftarrow BPA_{h_b}(h_b, H_f \cup (H_b \setminus H_b^{col}), P, w)))$ **then**
            $success \leftarrow FALSE$
            $H_f \leftarrow H_f \cup \{h_b\}$
            $H_b \leftarrow H_b \setminus \{h_b\}$
            restart
        else
            $M_{w_b} \leftarrow M_{w_b} \cup \{(h_b, w_b)\}$
    if $(\neg(KoordiniereBewegungen(M_{w_b})))$ **then**
        $success \leftarrow FALSE$
        choose $h_b \in H_b^{col}$
        $H_f \leftarrow H_f \cup \{h_b\}$
        $H_b \leftarrow H_b \setminus \{h_b\}$
        restart

In *Anhang B* wird die Vorgehensweise der Heuristik anhand von Beispielen veranschaulicht.

Ausblick

Die bekannten Bewegungsplanungsalgorithmen versuchen in der Regel aufgrund eines gegebenen Bewegungsplanungsproblems in der Szene eine Lösung für das Problem zu finden. Dabei nutzen sie keine Möglichkeiten, die Szene zu modifizieren, um dann darin schneller eine Lösung für das gestellte Problem zu berechnen. Insbesondere bei der Betrachtung großer Szenen sind dadurch keine Mechanismen gegeben, aus der Gesamtszene die für das Bewegungsplanungsproblem relevante Teilszene auszusuchen und darin das Problem zu lösen. Dadurch können in akzeptabler Laufzeit nur Bewegungsplanungsprobleme in relativ kleinen Szenarien gelöst werden.

Desweiteren werden durch die starre Einteilung der Objekte einer Szene in *bewegte Objekte* und *Hindernisse* der meisten bekannten Bewegungsplanungsalgorithmen mögliche Lösungen eines Bewegungsplanungsproblems nicht erkannt. Die Erweiterung der Bewegungsplanungsalgorithmen auf die gleichzeitige Betrachtung mehrerer bewegter Hindernisse führt zwar zu dem theoretischen Resultat, daß das allgemeine Bewegungsplanungsproblem lösbar ist, bietet jedoch keinen praktikablen Ansatz zu dessen Lösung. Gleichzeitig ist es fraglich, ob durch die Verbesserung vollständiger Algorithmen zur Lösung des allgemeinen Bewegungsplanungsproblems ein praktikabler Algorithmus dafür gefunden werden kann.

Vielmehr erscheint es sinnvoll, durch heuristische Vorgehensweisen die Möglichkeiten der Bewegungsplanung zu erweitern und dabei den Verlust der Vollständigkeit der Algorithmen in Kauf zu nehmen. Ein bisher zuwenig beachtetes Potential an Bewegungsplanungsstrategien bietet dabei die Vorgehensweise eines Menschen bei der Lösung von Bewegungsplanungsproblemen. Da der Mensch täglich mit einer Vielzahl von Bewegungsplanungsproblemen konfrontiert ist, die er im Rahmen seiner geistigen Fähigkeiten vielfach in Realzeit zu lösen vermag, bietet sich die Möglichkeit, aus der Anschauung der menschlichen Vorgehensweise algorithmische Konzepte abzuleiten, die die Flexibilität menschlichen Denkens mit der Fähigkeit eines Computers verbinden, komplexe Probleme zu handhaben. Dadurch können Bewegungsplanungsalgorithmen gefunden werden, die die bestehenden Möglichkeiten der Bewegungsplanung erweitern.

Anhang A

Laufzeitergebnisse für die heuristisch inkrementelle Bewegungsplanungsstrategie

In der Folge werden die Laufzeitergebnisse für die Lösung von Bewegungsplanungsproblemen mit Hilfe der heuristisch inkrementellen Bewegungsplanungsstrategie und des in *Kapitel 3* beschriebenen klassischen Bewegungsplanungsalgorithmus vorgestellt. Dabei wird deutlich, daß durch die heuristisch inkrementelle Strategie in der Regel Laufzeiteinsparungen erfolgen, jedoch auch Beispiele existieren, bei denen eine Laufzeitverschlechterung in Kauf genommen werden muß, dadurch jedoch die Lösungsmöglichkeiten des klassischen Bewegungsplanungsalgorithmus nicht beeinflußt werden.

Die Messung der Laufzeiten der Algorithmen erfolgte mit der LEDA-Zeitmeßfunktion in CPU-Sekunden (vgl. [Nae92]).

A.1 Bewegung eines Stuhles in einer Etage

Ein Beispiel für eine Szene, in der die heuristisch inkrementelle Bewegungsplanungsstrategie zu einer Verbesserung der Gesamtlaufzeit gegenüber der klassischen Betrachtungsweise der gesamten Szene führt, ist die in Abbildung A.1 und A.2 veranschaulichte Problemstellung.

Die Vorteile der heuristisch inkrementellen Bewegungsplanungsstrategie sind:

1. Der Weg entsteht durch lokale Modifikationen der direkten Verbindung zwischen Start- und Endposition.
2. Die Beschreibung des kürzesten kollisionsfreien Weges wird nur von einer kleinen Anzahl von Hindernissen – im Vergleich zur Gesamtanzahl der Hindernisse in der Szene – bestimmt.

Das führt bei der Durchführung der heuristischen Bewegungsplanung dazu, daß

1. keine oder nur wenige Hindernisse während der Bewegungsplanung betrachtet werden, die den kürzesten kollisionsfreien Weg *nicht* bestimmen,
2. die Anzahl der Iterationen klein und die gesamte Anzahl von betrachteten Hindernissen gering ist im Vergleich zur Gesamtanzahl der Hindernisse in der Szene.

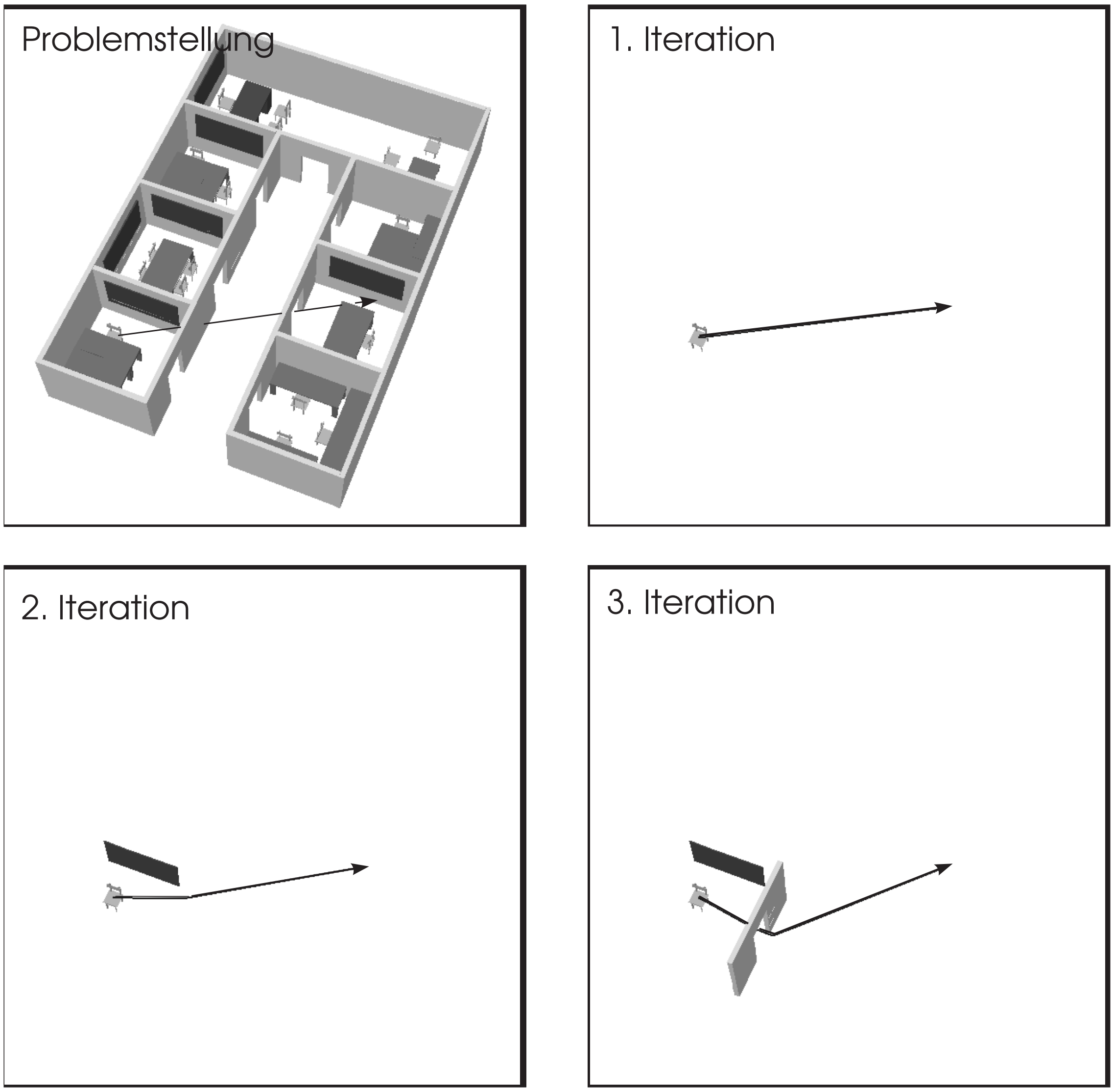

Abbildung A.1: Bewegung eines Stuhles in einer Etage und die Iterationen des Algorithmus, Teil 1

Als Varianten der heuristisch inkrementellen Bewegungsplanungsstrategie wurden die bereits in *Kapitel 2* erläuterten *minimal inkrementelle* und *inkrementelle* Strategie getestet.

Für das Bewegungsplanungsproblem ergaben sich folgende Laufzeiten für die betrachteten Rechnertypen:

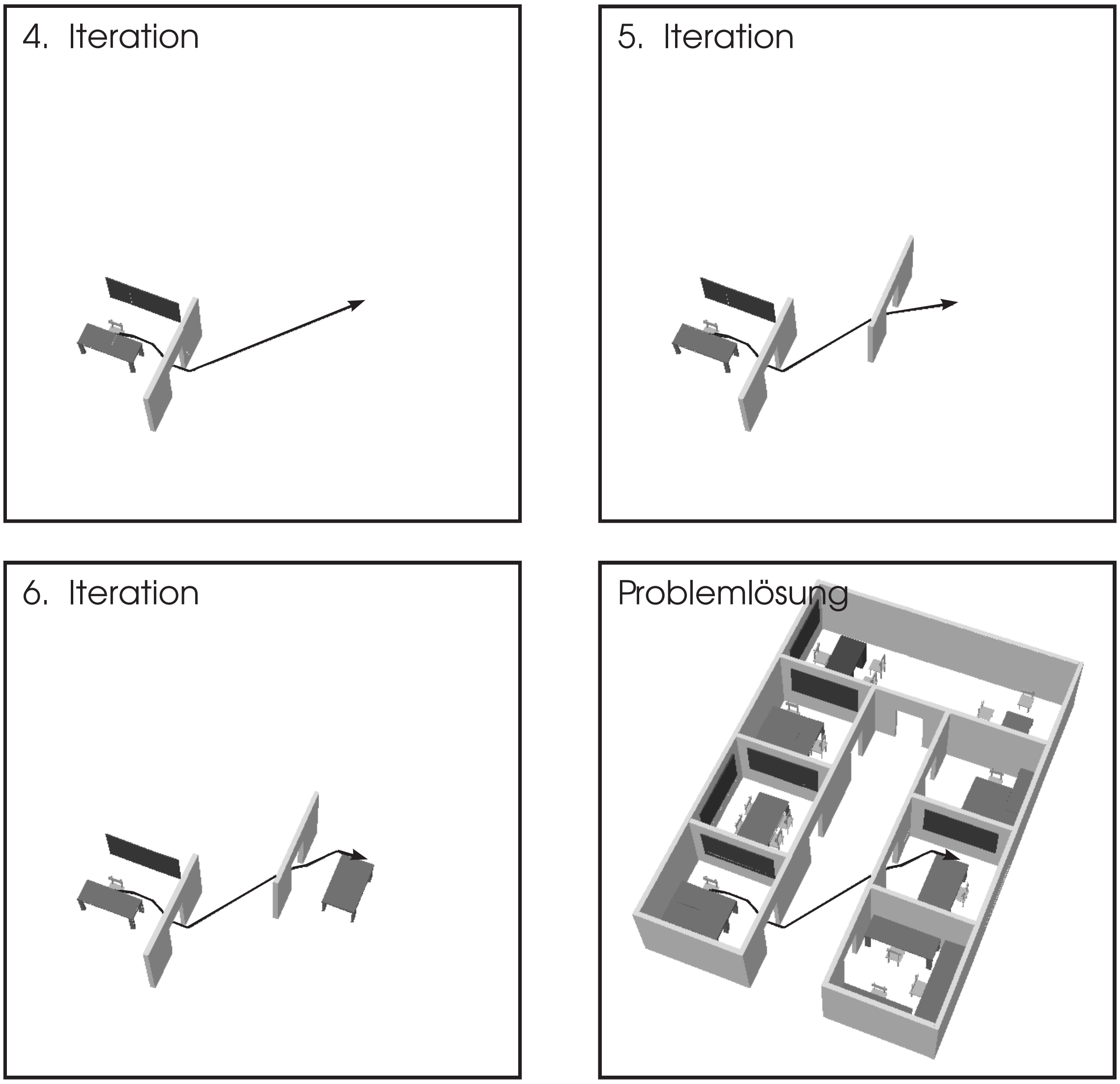

Abbildung A.2: Bewegung eines Stuhles in einer Etage und die Iterationen des Algorithmus, Teil 2

Vorgehensweise Rechnertyp	Min. inkrementell	inkrementell in [CPU-Sekunden]	klassisch	max. Laufzeitersparnis in [% von klassisch]
Sun SPARC 2	173.550	202.383	2807.866	93.82
Sun SPARC 10	90.167	117.750	1560.833	94.22
Sun SPARC 20	58.800	75.733	1012.267	94.19
HP 9000/730	75.790	94.100	1522.130	95.02
SGI Indy	90.760	115.190	1472.990	93.84

Die Laufzeitergebnisse sind in Abbildung A.3 graphisch veranschaulicht.

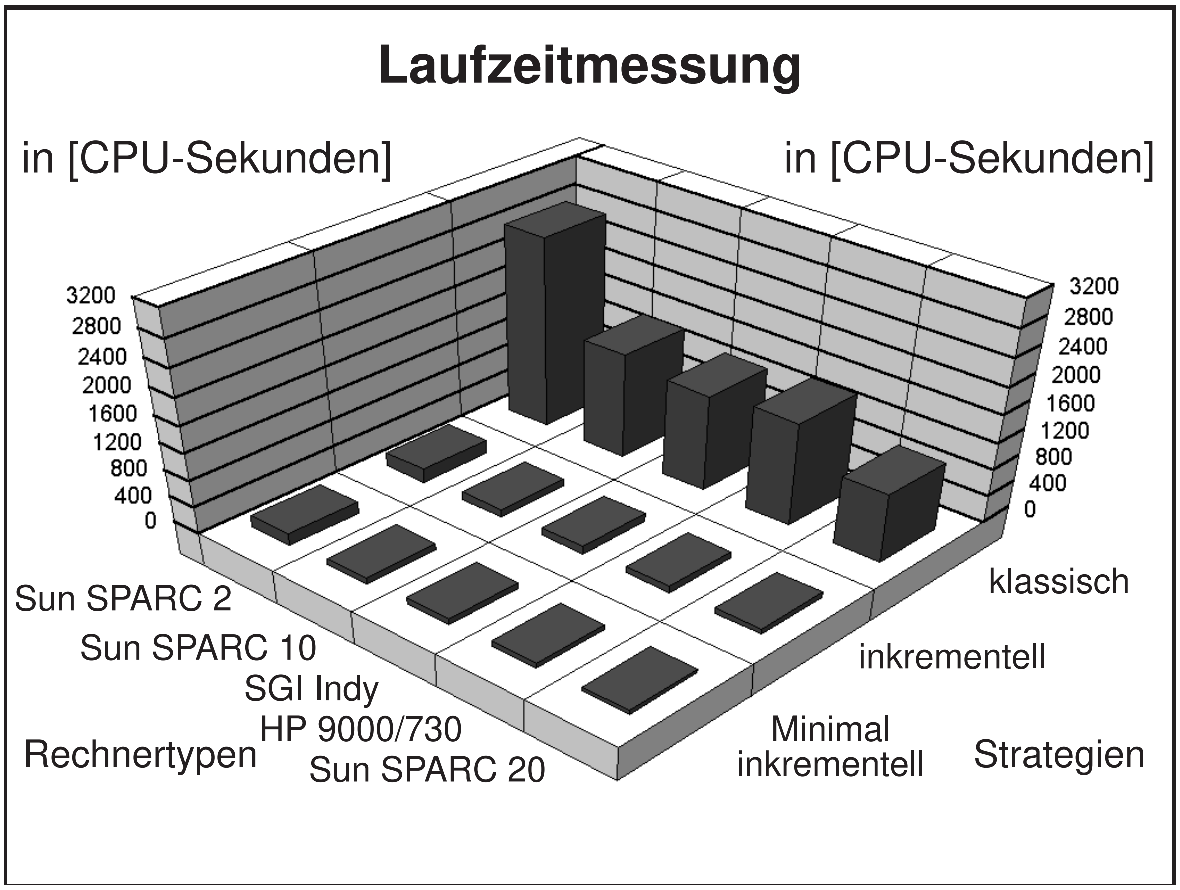

Abbildung A.3: Laufzeiten für die Bewegung eines Stuhles in einer Etage

A.2 Würfel umringt von Würfeln

Als Beispiel für eine Szene, in der die heuristisch inkrementelle Bewegungsplanung zu einer Verschlechterung der Gesamtlaufzeit gegenüber der klassischen Betrachtungsweise der gesamten Szene führt, kann die in der Abbildung A.5 angegebene Problemstellung angesehen werden. Das Problem für die heuristisch inkrementelle Bewegungsplanungsstrategie ist dabei, daß

1. alle Hindernisse zusammen den Wegeplan unmöglich machen,
2. in jeder Iteration durch die Positionierung der Hindernisse jeweils nur ein zusätzliches Hindernis in die Betrachtung aufgenommen wird.

Das führt in der Berechnung dazu, daß

1. die Folge der betrachteten Bewegungsplanungsprobleme $B_0, \ldots, B_n$ maximal lang wird,

2. die Unmöglichkeit der Bewegung erst in der letzten Iteration sichtbar wird.

Da erst in der letzten Iteration des Algorithmus erkannt wird, daß die Bewegung nicht möglich ist, erfolgt für alle vorhergehenden Iterationen die Berechnung eines Sichtbarkeitsgraphen, während bei der klassischen Vorgehensweise diese Berechnung entfällt.

Für dieses Beispiel ergaben sich folgende Laufzeiten für die verschiedenen Rechnertypen:

Vorgehensweise Rechnertyp	Min. inkrementell	inkrementell in [CPU-Sekunden]	klassisch	max. Laufzeitverlust in [% von klassisch]
Sun SPARC 2	21.100	22.083	9.633	129.24
Sun SPARC 10	11.000	11.183	4.767	134.59
Sun SPARC 20	7.033	7.417	3.017	145.84
HP 9000/730	9.320	9.460	4.060	129.56
SGI Indy	11.470	12.150	4.900	147.96

Die Laufzeitergebnisse sind in Abbildung A.4 graphisch dargestellt.

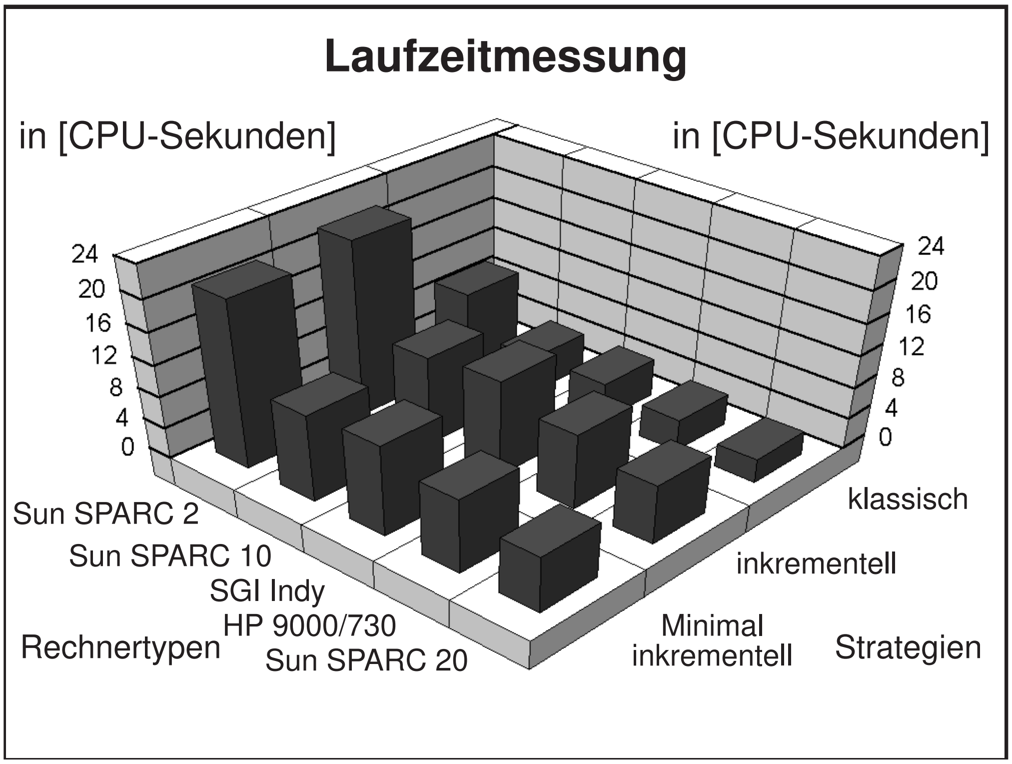

Abbildung A.4: Laufzeiten für einen Würfel umringt von Würfeln

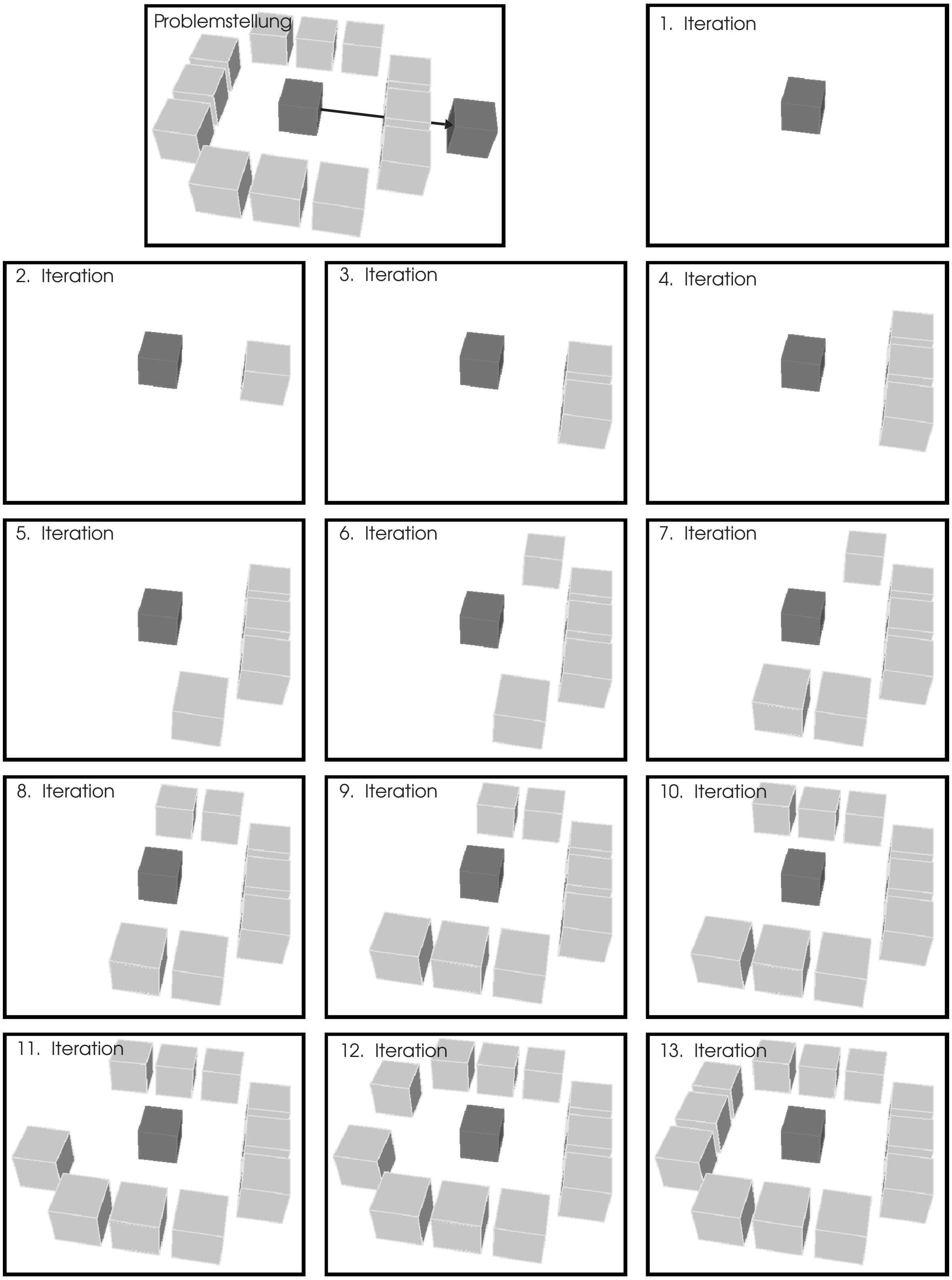

Abbildung A.5: Würfel umringt von 12 Würfeln und die Iterationen des Algorithmus

Anhang B

Beispiele für heuristische Bewegungsplanungsstrategie mit bewegbaren Hindernissen

In diesem Abschnitt wird die Vorgehensweise der heuristischen Bewegungsplanungsstrategie bei bewegbaren Hindernissen aus *Kapitel 4* anhand dreier Beispiele verdeutlicht. Dazu werden Bewegungsmöglichkeiten der Objekte entsprechend der Realisation des Bewegungsplanungsalgorithmus aus *Kapitel 3* angenommen, der im Rahmen dieser Arbeit im $\mathbb{IGOR}$ -System implementiert wurde. D.h. alle beweglichen Objekte in der Szene besitzen zwei translatorische Freiheitsgrade.

B.1 Bewegung eines Würfels in einem Gang

Als erstes Beispiel wird die Szene aus Abbildung 4.9 betrachtet. Das Ergebnis der gesamten Bewegungsplanung kann in Abbildung B.1 nachvollzogen werden. Die Vorgehensweise des Algorithmus bei der Berechnung dieser Lösung soll nun in der Folge näher erläutert werden.

In einem ersten Schritt wird für das bewegte Objekt P ohne Berücksichtigung der bewegbaren Hindernisse h_b^1 und h_b^2 ein Weg mit dem in *Kapitel 3* beschriebenen Algorithmus geplant. Es ergibt sich dabei die in Abbildung B.1 (Bild *Problemlösung*) dargestellte Bewegung.

In einem zweiten Schritt wird mit Hilfe der Kollisionsrechnung erkannt, daß die beiden bewegbaren Hindernisse h_b^1 und h_b^2 die berechnete Bewegung für P behindern.

Im dritten Schritt wird nacheinander für h_b^1 und h_b^2 jeweils eine individuelle Straßenkarte von Ausweichbewegungsmöglichkeiten berechnet. Diese bestehen aus den jeweiligen Sichtbarkeitsgraphen und sind in Abbildung B.2 zu sehen. Aus dem in Abbildung B.3 zu dargestellten, für h_b^1 berechneten Konfigurationsraum ist zu erkennen, daß die weiße Region der freien Positionen bzgl. der Bewegung von P für h_b^1 nur in einer Nische erreicht werden kann. Analog ergibt sich als freie Positionen für h_b^2 bzgl. der Bewegung von P eine Region in der anderen Nische. Somit ergeben sich keine Konflikte bei der Wahl der Zielpositionen für h_b^1 und h_b^2, jedoch Koordinationsprobleme für die Ausweichbewegungen.

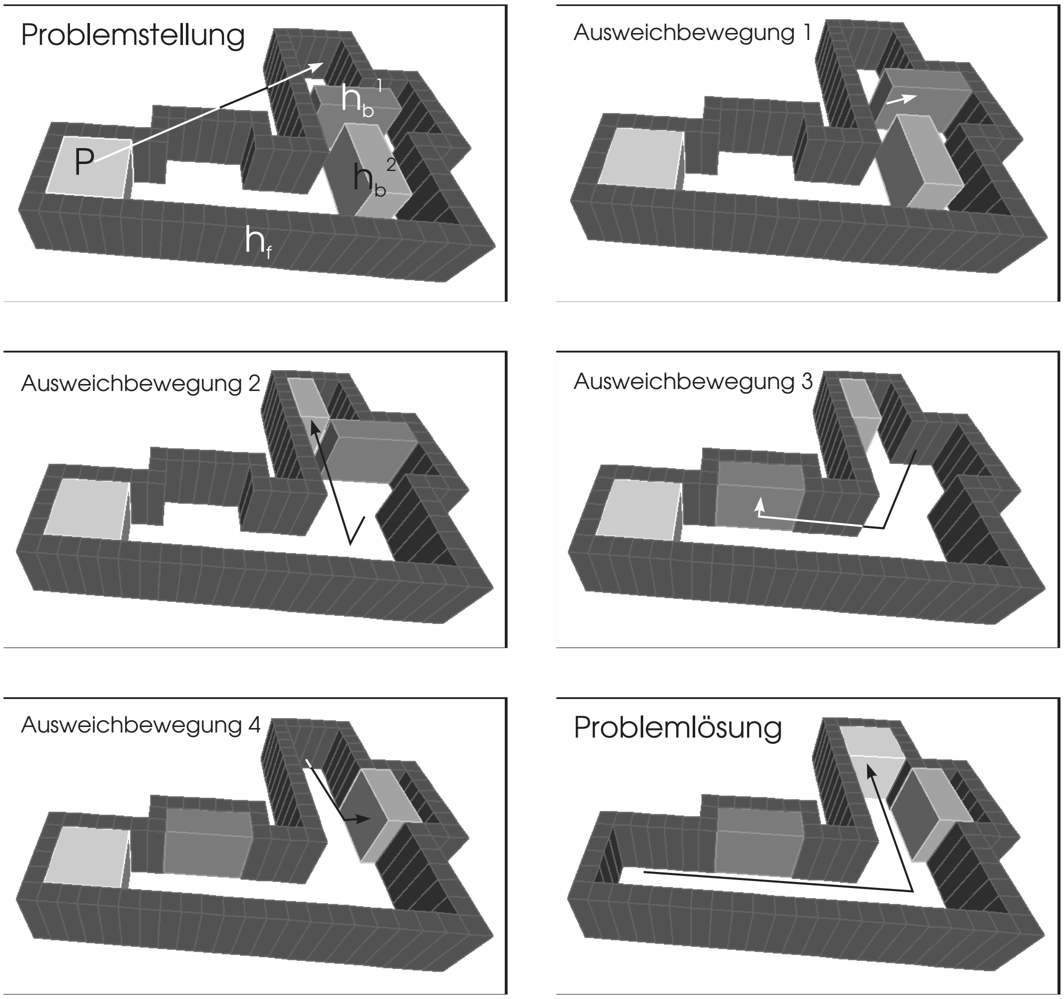

Abbildung B.1: Bewegung eines Würfels in einem Gang mit bewegbaren Hindernissen

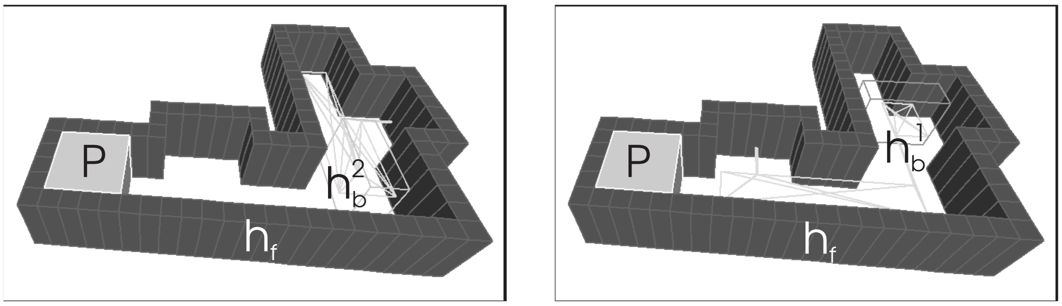

Abbildung B.2: Straßenkarten für die bewegbaren Hindernisse

Wie in Abbildung B.1 (Bilder *Ausweichbewegung 1 – Ausweichbewegung 4*) zu sehen ist, muß

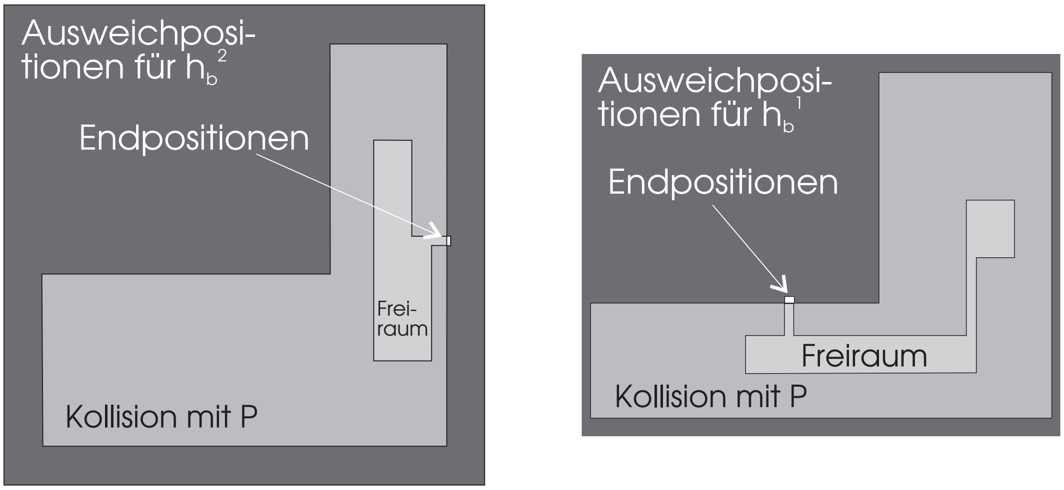

Abbildung B.3: Konfigurationsräume für h_b^1 und h_b^2

zur Bewegungskoordination der Ausweichbewegungen für h_b^1 und h_b^2 jede der beiden einzelnen Ausweichbewegungen unterbrochen werden, um eine gesamte Koordination der Ausweichbewegungen zu erreichen. Eine Berechnung und Koordination der Ausweichbewegungen mit Hilfe des Algorithmus von Erdmann und Lozano-Perez [ELP87] hätte nicht zum Ziel geführt, da dieser jeweils immer eine vollständige Bewegung für ein Objekt plant und diese Bewegung nachträglich nicht wieder modifiziert. Mit Hilfe des Koordinationsgraphen und der Berechnung der Kanten unter Vernachlässigung des Zeitaspektes für die Einzelbewegungen wird die Abfolge der Ausweichbewegungen gefunden.
Damit ist das gesamte Bewegungsplanungsproblem, wie in Abbildung B.1 zusammengefaßt, gelöst.

B.2 Würfel umringt von Würfeln

An dieser Stelle soll das in Abschnitt A.2 angegebene Beispiel für die heuristisch inkrementelle Bewegungsplanungsstrategie erneut aufgegriffen werden und dessen Lösbarkeit bei Berücksichtigung bewegbarer Hindernisse betrachtet werden (vgl. Abbildung B.4).

Dabei ist es nicht wichtig, ob einer, mehrere oder alle Hinderniswürfel in der Szene bewegbar sind. Die Lösung des Problems erfolgt immer in analoger Art und Weise.

Durch die Berechnung einer kollisionsfreien Bewegung für den bewegten Würfel unter Berücksichtigung nur der festen Hinderniswürfel wird ein Weg gefunden, wie z.B. in Abbildung B.5 veranschaulicht.

Dieser Weg kollidiert mit genau einem Hinderniswürfel, für den eine geradlinige Ausweichbewegung berechnet werden kann (vgl. Abbildung B.6).

Eine Koordination von Ausweichbewegungen entfällt.

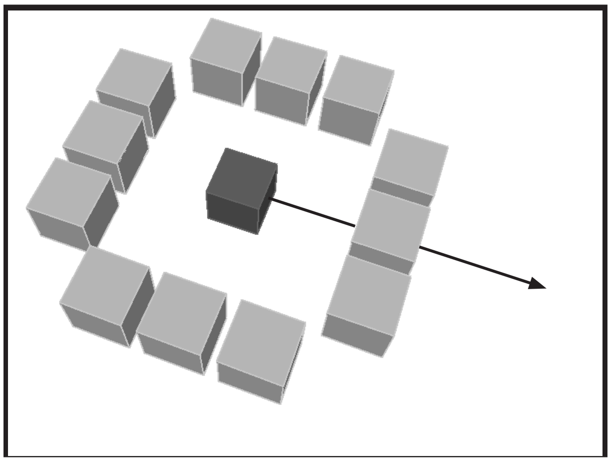

Abbildung B.4: Würfel umringt von 12 Würfeln

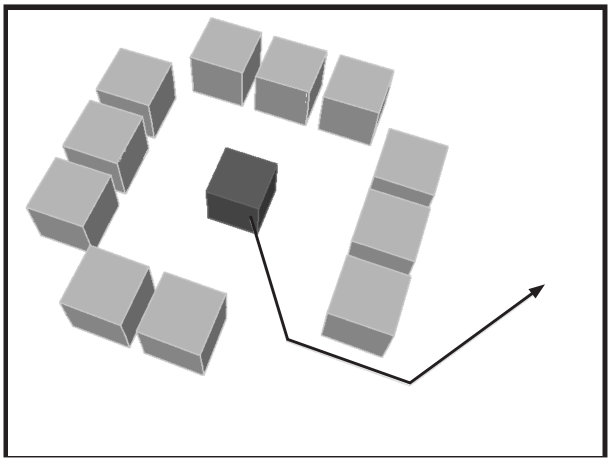

Abbildung B.5: Kollisionsfreie Bewegung eines Würfels

Damit ist das Bewegungsplanungsproblem, das mit Hilfe der klassischen Bewegungsplanung nicht lösbar ist, bei Berücksichtigung bewegbarer Hindernisse jedoch intuitiv einfach ist, mit Hilfe der Heuristik in einfacher Weise gelöst. Die Lösung ist in Abbildung B.7 zusammengefaßt.

B.3 Bewegung eines Schreibtisches in einer Etage

In der bereits in *Anhang A* betrachteten Szene kann, wie in Abbildung B.8 (Bild Problemstellung) veranschaulicht, ein Bewegungsplanungsproblem spezifiziert werden, dessen Lösung

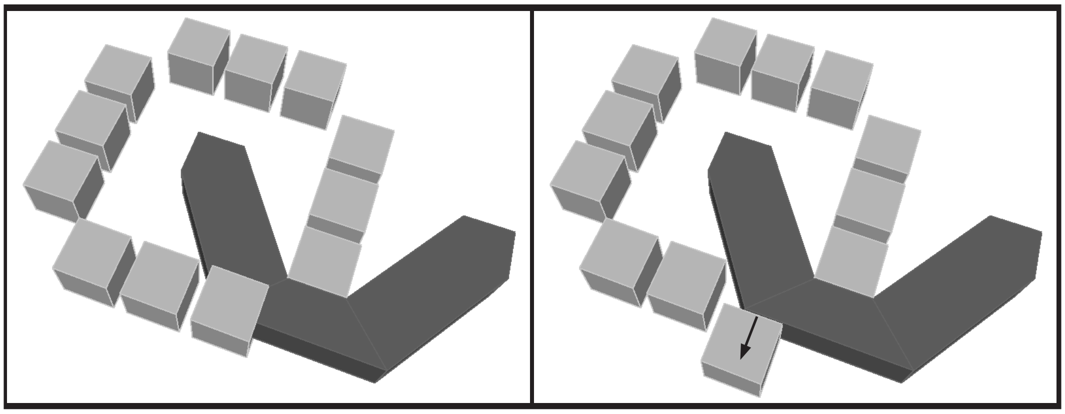

Abbildung B.6: Kollisionsfreie Ausweichbewegung eines Hinderniswürfels

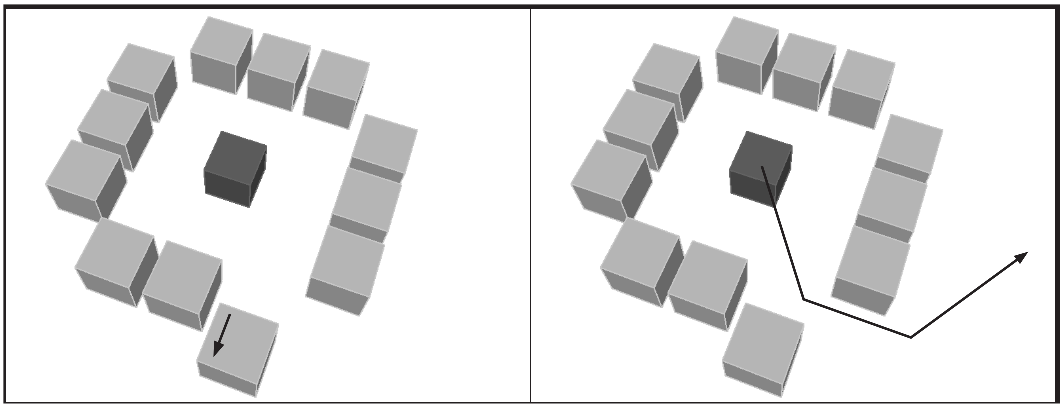

Abbildung B.7: Gesamte Bewegungsplanung für einen Würfel umringt von Würfeln

intuitiv einfach ist, mit Hilfe des klassischen Bewegungsplanungsalgorithmus von *Kapitel 3* jedoch nicht berechnet werden kann.

Bei der Verwendung der heuristischen Bewegungsplanungsstrategie ohne zeitliche Koordination der Ausweichbewegungen der bewegbaren Hindernisse kann das Problem bei sukzessiver Bewegung der störenden, bewegbaren Hindernisse gelöst werden, wenn z.B. alle Hindernisse in der Szene außer den Wänden als bewegbar klassifiziert werden.
Dabei können für die beiden störenden, bewegbaren Hindernisse einfache Ausweichbewegungen (vgl. Abbildung B.8 (Bilder *Ausweichbewegung 1* und *Ausweichbewegung 2*)) berechnet werden, die in jeder der beiden möglichen Reihenfolgen durchlaufen werden können. Dadurch wird die einfache Bewegung für das zu bewegende Objekt ermöglicht.

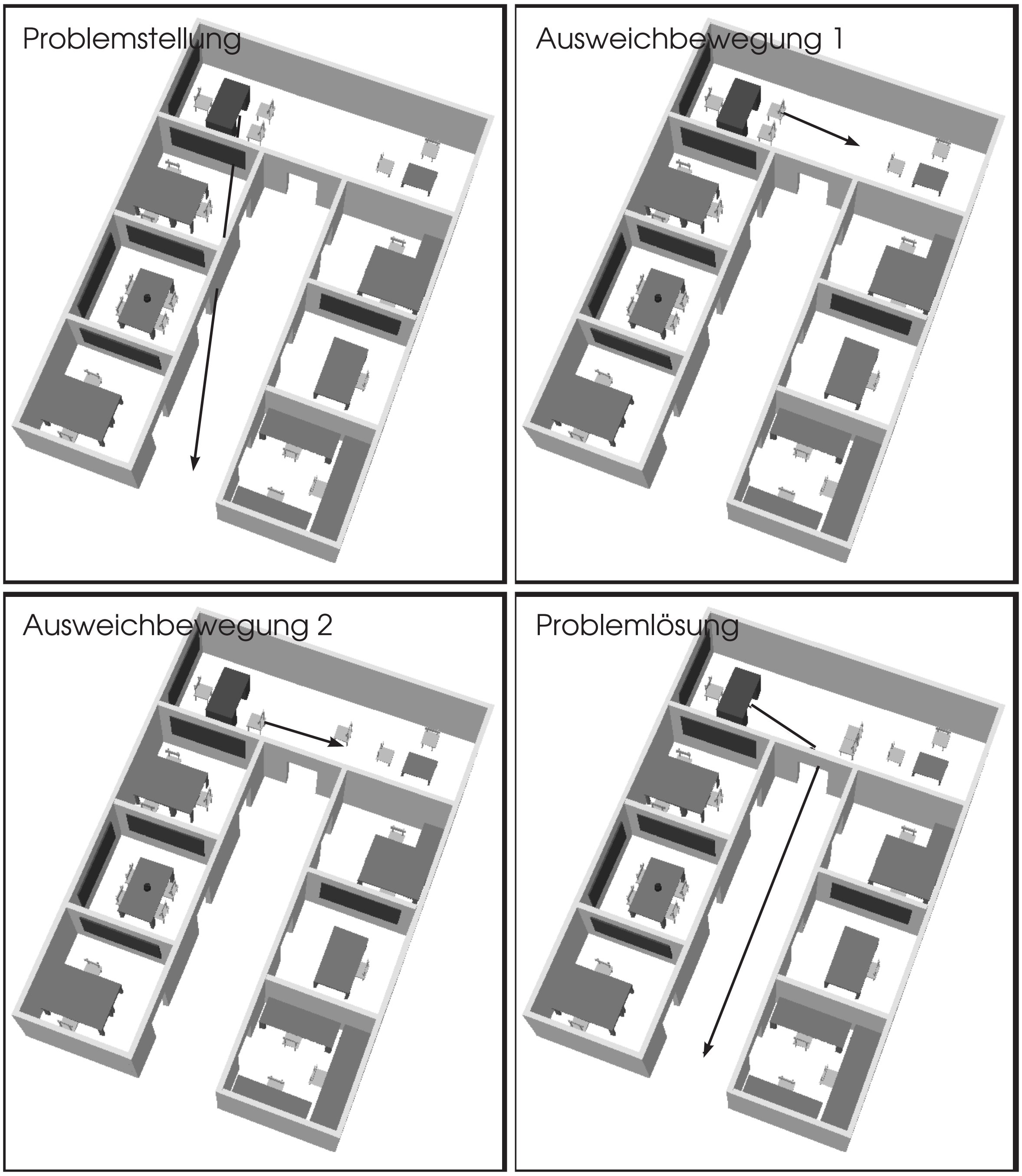

Abbildung B.8: Bewegung eines Schreibtisches in einer Etage

Anhang C

Das IGOR-System

Der in *Kapitel 3* beschriebene Algorithmus wurde zusammen mit der Erweiterung um die heuristisch inkrementelle Bewegungsplanungsstrategie im Rahmen dieser Arbeit innerhalb des IGOR-Systems realisiert. Ihm entstammen die Laufzeitmessungen aus *Abschnitt A*.

Das IGOR-System ist ein graphisches System zur interaktiven Robotersimulation, Bewegungs- und Montageplanung. Es besteht aus den Komponenten:

1. **Visualisierung und interaktive Bewegungsspezifikation**

   Das IGOR-Kernsystem, das im Rahmen von [Sch94] implementiert wurde, erlaubt die graphische Darstellung von 3-dimensionalen Objekten, die Festlegung der Perspektive, die interaktive Angabe von Bewegungen durch eine Folge von Translationen und Rotationen sowie deren Animation.

2. **Kollisionsrechnung**

   Alle im IGOR-System spezifizierten Bewegungen können mit Hilfe der im Rahmen von [Fr93] implementierten Kollisionsrechnung auf Kollisionen getestet werden. Dieses Modul ist eine direkte Erweiterung des IGOR-Kernsystems und wird von allen nachfolgenden Modulen benutzt.

3. **Robotersimulation**

   Im Rahmen von [Si94] wurde ein System zur interaktiven Robotersimulation und Montageplanerstellung realisiert, das die Generierung von Roboterprogrammen ermöglicht, wobei die Kollisionsfreiheit der Roboterprogramme mit Hilfe der Kollisionsrechnung verifiziert wird.

4. **Bewegungsplanung**

   Zur automatischen Berechnung von kollisionsfreien Bewegungen für Objekte im Raum wurden im Rahmen von [Ch95] und dieser Arbeit zwei Bewegungsplanungsalgorithmen implementiert. Dabei erfolgt die Spezifikation des Bewegungsplanungsproblems interaktiv. Der im Rahmen dieser Arbeit implementierte Algorithmus ist in *Kapitel 3* beschrieben. Bei der Erweiterung um die heuristisch inkrementelle Bewegungsplanungsstrategie wird die Kollisionserkennung in die Bewegungsplanung einbezogen.

> Bei dem im Rahmen von [Ch95] realisierten Bewegungsplanungsalgorithmus handelt es sich um ein heuristisches Verfahren zur Berechnung von translatorischen und rotatorischen Bewegungen für ein Objekt.

Die Implementierung des Systems basiert auf dem Betriebssystem *Unix*, der graphischen Oberfläche *X-Windows* des MIT und der Oberflächengestaltung *OSF/Motif*. Die Module sind in der Programmiersprache *C++* implementiert unter Verwendung der Software-Bibliothek *LEDA* des Max-Planck-Institutes für Informatik, Saarbrücken.

Die Benutzeroberfläche des Systems ist in Abbildung C.1 veranschaulicht. Sie besteht im Falle der Bewegungsplanung aus den vier Komponenten

- Graphische Darstellung der Objekte im Objektraum (großes Fenster)
- Interaktive Spezifikation von Start- und Zielposition mit Hilfe des Trackballs (kleines Fenster rechts oben)
- Graphische Darstellung des berechneten Konfigurationsraumes (Fenster rechts Mitte)
- Laufzeitinformationen des Algorithmus (Fenster rechts unten)

Dabei wird mit Hilfe des Trackball-Fensters ein Bewegungsplanungsproblem im Objektraum spezifiziert, das nach der Angabe einer Bewegungsebene mit Hilfe des Bewegungsplanungsalgorithmus gelöst wird. Die Problemtransformation vom Objektraum in den durch den Algorithmus berechneten Konfigurationsraum kann graphisch veranschaulicht werden. Nach der Lösung des Problems kann der berechnete Weg im Konfigurations- und im Objektraum dargestellt werden und die Bewegung mit Hilfe des IGOR-Kernsystems animiert und mit Hilfe der Kollisionsrechnung auf Kollisionsfreiheit gestestet werden.

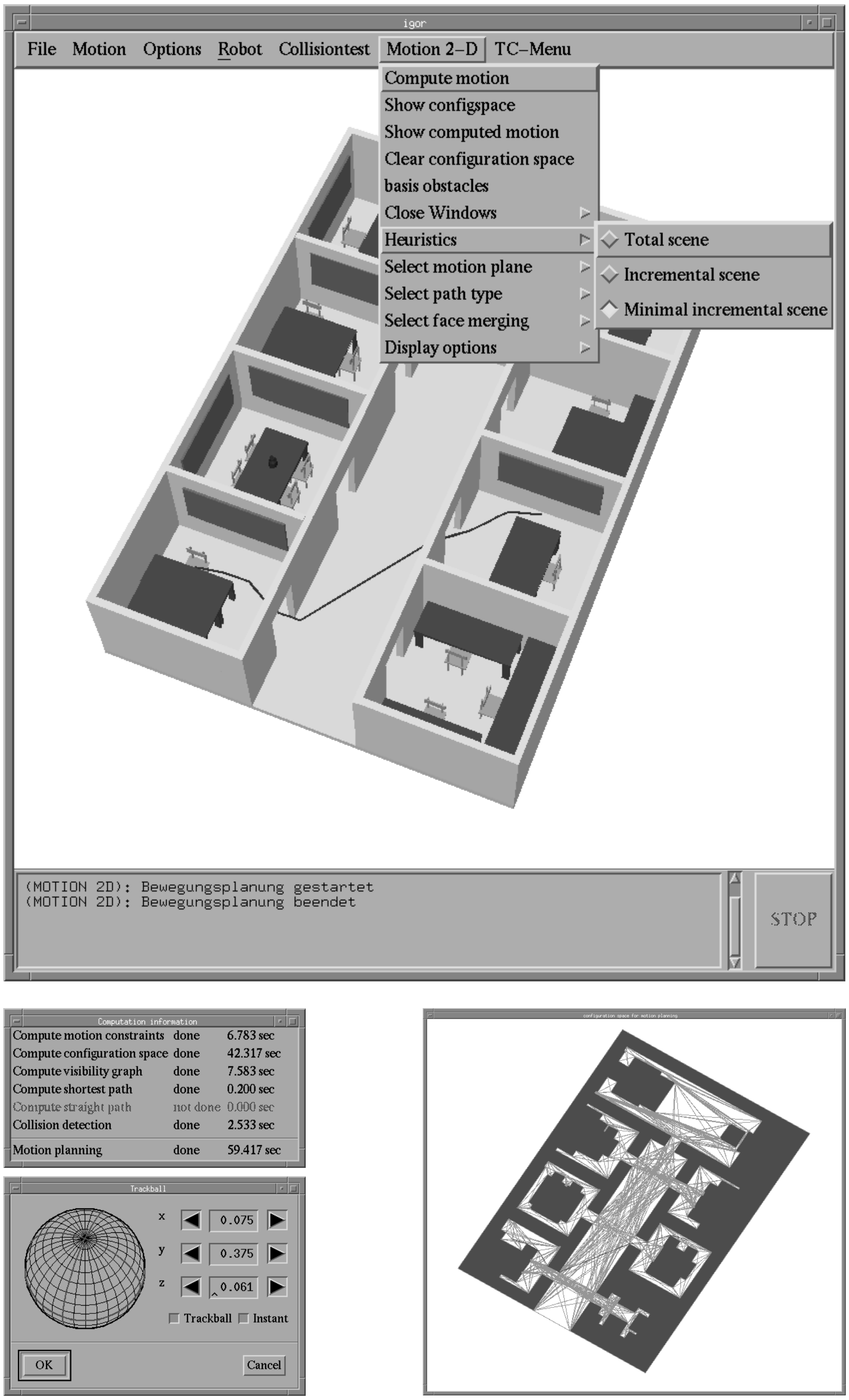

Abbildung C.1: Graphische Benutzeroberfläche des Bewegungsplanungsalgorithmus im IGOR-System

Literaturverzeichnis

[AB88] F. Avnaim, J.D. Boissonnat
Polygon Placement Under Translation and Rotation
Technical Report No. 889, INRIA, Sophia-Antipolis, France

[AS93] B. Aronov, M. Sharir
The Union of Convex Polyhedra in Three Dimensions
Proceeding of the 34th IEEE Symposium on Foundations of Computer Science, S. 518–527, 1993

[BO79] J.L. Bentley, T.A. Ottmann
Algorithms for Reporting and Counting Geometric Intersections
IEEE Transactions on Computers, Vol. C-28, S. 643–647, 1979

[Boy79] J.W. Boyse
Interference Detection Among Solids and Surfaces
Communications of the ACM, Vol. 22, No. 1, S. 3–9, 1979

[Ca87] J.F. Canny
The Complexity of Robot Motion Planning
ACM Doctoral Dissertation Award 1987, MIT Press, 1987

[CEG+93] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, J. Snoeyink
Computing a Face in an Arrangement of Line Segments
SIAM Journal of Computing, Vol. 22, S. 1286–1302, 1993

[Ch95] T. Chadzelek
Heuristische Bewegungsplanung mit vielen Freiheitsgraden
Diplomarbeit an der Universität des Saarlandes, 1995

[Col75] G. E. Collins
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition
Proceeding of the 2nd GI Conference on Automata Theory and Formal Languages, S. 134–183, Springer Verlag LNCS 33, 1975

[CP90] B. Chazelle, L. Palios
Triangulating a Nonconvex Polytop
Discrete Comput. Geom., Vol. 5, S. 505–526, 1990

[DK85] D.P. Dopkin, D.G. Kirkpatrick
A Linear Algorithm for Dertermining the Separation of Convex Polyhedra
J. Algor., Vol. 6, S. 381–392, 1985

[DO92] A. Dhagat, J. O'Rourke
Motion Planning Amidst Movable Square Blocks
Proceeding of 4th Canadian Conference on Computational Geometry, S. 188–191, 1992

[Ed87] H. Edelsbrunner
Algorithms in Combinatorial Geometry
Springer Verlag, 1987

[ELP87] M. Erdmann, T. Lozano-Perez
On Multiple Moving Objects
Algorithmica, Vol. 2, S. 477–521, 1987

[ES93] J. Eckstein, A. Schacke
Bewegungsplanung mit zwei Freiheitsgraden
Fortgeschrittenenpraktikum an der Universität des Saarlandes, 1993

[Fr93] M. Fritz
Kollisionstests in der Objektmontage
Diplomarbeit an der Universität des Saarlandes, 1993

[Fu91] K. Fujimura
Motion Planning in Dynamic Environments
Springer-Verlag, 1991

[GM87] S.K. Ghosh, D.M. Mount
An Output Sensitive Algorithm for Computing Visibility Graphs
Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, S. 11–19, 1987

[GS86] L. Guibas, R. Seidel
Computing Convolution by Reciprocal Search
Proceedings of the ACM Symposium on Computational Geometry, S. 90–99, 1986

[GSS89] L.J. Guibas, M. Sharir, S. Sifrony
On the General Motion-Planning Problem with Two Degrees of Freedom
Discrete Computational Geometry, Vol. 4, S. 491–521, 1989

[HS93] J. Hershberger, S. Suri
Efficient Computation of Euclidean Shortest Paths in the Plane
Proceeding 34th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93), S. 508–517, 1993

[KS88] K. Kedem, M. Sharir
An Automatic Motion Planning System for a Convex Polygonal Mobile Robot in 2-d Polygonal Space
Proceeding of the 4th ACM Symposium on Computational Geometry, S. 329–340, 1988

[KZ86] K. Kant, S.W. Zucker
Toward efficient trajectory planning: The Path-Velocity Decomposition
International Journal of Robotics Research, Vol. 5, S. 72–89, 1986

[La91] J.-C. Latombe
Robot Motion Planning
Kluwer Academic Publishers, 1991

[LP83] T. Lozano-Perez
Spatial Planning: A Configuration Space Approach
IEEE Transactions on Computers, Vol. 32, S. 108–120, 1983

[LPW79] T. Lozano-Perez, M. Wesley
An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles
Communications of the ACM, Vol. 22, S. 560–570, 1979

[MN94] K. Mehlhorn, S. Näher
Implementation of a Sweep Line Algorithm for the Straight Line Segment Intersection Problem
Technischer Bericht, MPI-I-94-160, Max-Planck-Institut für Informatik, Saarbrücken, 1994

[Nae92] S. Näher
LEDA *User Manual: Library of Efficient Data Types and Algorithms*
Version 3.0, Max-Planck-Institut für Informatik, Saarbrücken, 1992

[NM65] J.A. Nelder, R. Mead
Computer Journal, Vol. 7. S. 305–309, 1965

[Pe84] J. Pearl
Heuristics
Addison-Wesley, 1984

[PS85] F.P. Preparata, M.I. Shamos
Computational Geometry
Springer Verlag, 1985

[SA84] M. Sharir, E. Ariel-Sheffi
On the 'Piano Movers' Problem: IV. Various Decomposable Two-dimensional Motion Planning Problems
Communications on Pure and Applied Mathematics, Vol. 37, S 479–493, 1984

[Sch92] S. Schirra
Approximative Bewegungsplanungsverfahren
Dissertation an der Universität des Saarlandes, 1992

[Sch92a] A. Schweikhard
A Simple Path Search Strategy Based on Calculation of Free Sections of Motions
Engn. Applic. Artif. Intell. Vol. 5, S. 1–10, 1992

[Sch94] E. Schömer
Interaktive Montageplanung mit Kollisionserkennung
Dissertation an der Universität des Saarlandes, 1994

[Si94] M. Sinnwell
Simulation von Roboterbewegungen mit On-Line Kollisionserkennung
Diplomarbeit an der Universität des Saarlandes, 1994

[SS83a] J.T. Schwartz, M. Sharir
On the 'Piano Movers' Problem. I. The Case of a Two-dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers
Communications on Pure and Applied Mathematics, Vol. 36, S. 345–398, 1983

[SS83b] J.T. Schwartz, M. Sharir
On the 'Piano Movers' Problem. II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds
Advances in Applied Mathematics, Vol. 4, S. 298–351, 1983

[SS83c] J.T. Schwartz, M. Sharir
On the 'Piano Movers' Problem. III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers
International Journal of Robotics Research, Vol 2. S. 46–75, 1983

[SS84] J.T. Schwartz, M. Sharir
On the 'Piano Movers' Problem: V. The Case of a Rod Moving in Three-dimensional Space Amidst Polyhedral Obstacles
Comunications on Pure and Applied Mathematics, Vol. 37, S. 815–848

[ST94] E. Schömer, Ch. Thiel
Efficient Collision Detection for Moving Polyhedra
Technischer Bericht, MPI-I-94-147, Max-Planck-Institut für Informatik, Saarbrücken, 1994

[Ud77] S. Udupa
Collision detection and avoidance in computer-controlled manipulators
Proceeding of IJCAI, 1977

[We85] E. Welzl
Constructing the Visibility Graph for n-Line Segments in $O(n^2)$ Time
Information Processing Letters, Vol. 20, S. 167–171, 1985

[Wi91] G. Wilfong
Motion Planning in the Presence of Movable Obstacles
Annals of Mathematics and Artificial Intelligence, Vol. 3, S. 131–150, 1991