Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Daniel Plohmann

Planung von kürzesten, sicheren Pfaden in Umgebungen mit unvorhersehbar beweglichen Hindernissen

12. Dezember 2007

Seminararbeit im WS 2007/2008

Zusammenfassung

Inhalt dieser Studienarbeit ist die Vorstellung eines Algorithmus zur Planung von bezüglich der Zeit kürzesten Wegen, die kollisionsfrei an sich bewegenden Hindernissen vorbei von einem Start- zu einem Zielpunkt führen.

Als Vorlage dient die Veröffentlichung "Planning the Shortest Safe Path amidst Unpredictably Moving Obstacles" [Berg06] von Jur van den Berg und Mark Overmars von der Universität Utrecht, aus dem Jahr 2006.

Inhaltsverzeichnis

1 Einführung

Bewegungsplanung zählt zu den maßgebenden Problemstellungen in der Robotik. Anhand derzeitiger Modellansätze bildet die Kollisionsvermeidung dabei das Fundament für darauf aufbauende Systeme und nimmt somit eine sehr wichtige Stellung ein. Bewegungen und nicht statische Hindernisse erhöhen die Komplexität der Aufgabenstellung enorm, oft lassen sich Pfade nur probabilistisch mit Hilfe von Sensorik und Prädiktion von globalen Zuständen planen.

Da sich die Welt kontinuierlich verändert, die Reaktionen aber nur auf diskreten Weltzuständen beruhen, kann es vorkommen, dass die angenommene Sicherheit eines Pfades durch plötzliche Änderungen und die veraltete Grundlage ungültig wird.

Um diesem Problem zu begegnen wird oftmals ein gewisses Zeitintervall τ für die Planung reserviert. Berechnungen finden dann mit dem geschätzten Zustand $t + \tau$ statt, was jedoch Schwierigkeiten beherbergt:

1. Wahrer Zustand und vorhergesagter Zustand können sich durch spontane Geschwindigkeitsänderungen stark unterscheiden, was in ungültigen Pfaden resultiert.

2. Der befahrene Pfad zwischen t und $t + \tau$ kann nicht als vollkommen sicher angenommen werden, da er auf alten Messdaten berechnet wurde.

Der hier vorgestellte Algorithmus ist fähig, dies zu vermeiden, hat jedoch die Voraussetzung, dass die maximale Geschwindigkeit der Hindernisse bekannt ist.

Konkret werden die beiden oben aufgeführten Probleme gelöst, indem

1. der gesamte Weltzustand zum Zeitpunkt $t + \tau$ in die Berechnung eingenommen wird und

2. der berechnete Pfad bewiesen kollisionsfrei bleibt, unabhängig des Verhaltens der Hindernisse.

2 Modell der Umgebung

2.1 Beschreibung des Modells

Wir können annehmen, dass sowohl der Roboter wie auch die Hindernisse vereinfacht in Form von umschließenden Kreisscheiben in der Ebene existieren. Dieses Modell wird im Konfigurationsraum noch vereinfacht: Der

Roboter wird auf einen Punkt reduziert, indem sein Radius zu dem der Kreisscheiben der Hindernisse addiert wird.

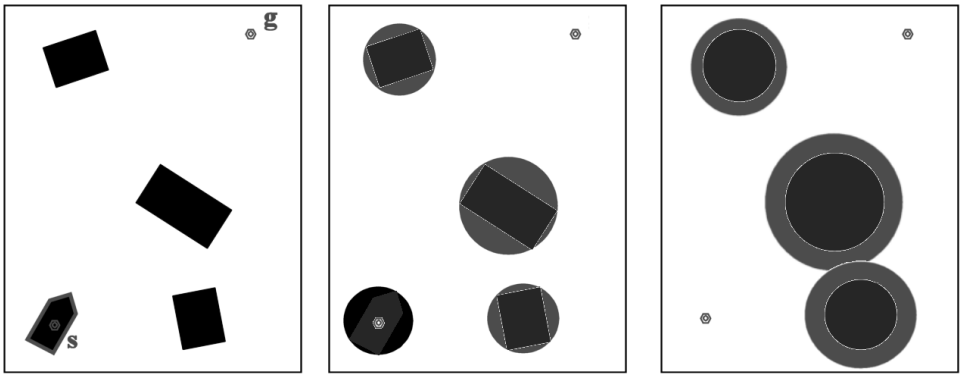

Abbildung 1: Von links nach rechts: Hindernisse und Roboter werden von Kreisscheiben umgeben. Dann wird der Radius des Roboters auf den der Kreisscheiben addiert, um einen für das Modell punktförmigen Roboter zu erhalten.

Des weiteren sollte der Roboter eine höhere Maximalgeschwindigkeit besitzen als die Hindernisse. Über den Verlauf der Zeit werden die Scheiben der Hindernisse als linear wachsend mit deren maximaler Geschwindigkeit betrachtet und als Kollisionsräume angesehen. Die Aufgabenstellung lautet somit: Finde einen bezüglich der Zeit kürzesten Pfad von Startpunkt s zum Zielpunkt t, ohne dabei eine der Kreisscheiben zu schneiden.

2.2 Mathematische Definition der Problemstellung

Gegeben sind n sich bewegende Objekte $O_1, ..., O_n$ in der Ebene, festgelegt durch ihren Kreismittelpunkt $p_i \in R^2$ und Anfangsradius $r_i \in R^+$. Die maximale Geschwindigkeit der Hindernisse ist gegeben als $v_i \in R^+$, wobei $i \in 1, ..., n$.

Der Roboter wird punktförmig angenommen, kann sich höchstens mit Geschwindigkeit $V \in R^+$ bewegen, wobei $V \geq v_i \forall i \in 1, ..., n$ gilt, startet bei $s \in R^2$ und hat den Zielpunkt $g \in R^2$.

Die Hindernisse müssen wir als offene Kreisscheiben $B(p, r) \subset R^2$ annehmen, p als Mittelpunkt und r als Radius, da sie sich zum Zeitpunkt t an unbekannter Stelle aufhalten und des weiteren keinerlei Information über

Richtung oder Geschwindigkeit bekannt ist.

Definition 1
Ein Punkt $p \in R^2$ ist kollisionsfrei zum Zeitpunkt $t \in R^+$, falls $p \notin \bigcup_i B(p_i, r_i + v_i t)$.

3 Kürzeste Pfade

3.1 Eigenschaften kürzester Pfade

Um einen kürzesten Pfad zu finden, zeigen wir zunächst einige Voraussetzungen auf, die sich bei den weiteren Betrachtungen als sehr hilfreich erweisen werden.

Das vorliegende Problem können wir auch in der dritten Dimension betrachten, was gewisse Vorteile mit sich bringt. Ein Punkt im Raum wird zu seinen unveränderten (x, y)-Koordinaten um die Zeit t als dritte Komponente erweitert.
Die Zeitachse verläuft orthogonal zu der gegebenen Ebene durch den Ursprung und hat dort die Zeit $t = 0$, dadurch werden die bereits eingeführten, sich ausdehnenden Kreisscheiben zu Kegeln im Raum. Dabei befindet sich die Kegelspitze auf der Senkrechten durch den Kreismittelpunkt pi, die Geschwindigkeit vi gibt den Öffnungswinkel an.
Formal ergibt sich die Formel des Kegels als:

$$C_i : (x - p_{ix})^2 + (y - p_{iy})^2 = (v_i t + r_i)^2$$

3.2 Bedeutung maximaler Geschwindigkeit

In diesem Abschnitt widmen wir uns der Betrachtung der Geschwindigkeit mit Hinblick auf kürzeste Pfade.

Lemma 2 *Ein Punkt $p \in R^2$, der zu einem Zeitpunkt $t = t'$ kollisionsfrei ist, ist auch zu allen $0 \leq t' \leq t$ kollisionsfrei.*

Beweis. Aus $t_1 \leq t_2$ können wir schließen, dass $B(p_i, r_i + v_i t_1) \subset (p_i, r_i + v_i t_2)$. Wenn nun also ein Punkt zum Zeitpunkt t_2 kollisionsfrei ist, so wird er auch zum Zeitpunkt t_1 kollisionsfrei sein, da die Kreisscheiben geringere Ausdehnung besitzen:
$p \notin \bigcup_i B(p_i, r_i + v_i t_2) \Rightarrow p \notin \bigcup_i B(p_i, r_i + v_i t_1)$ □

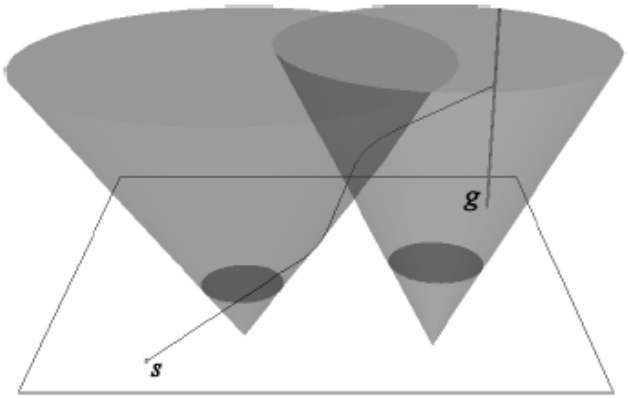

Abbildung 2: Visualisierung der dreidimensionalen Auffassung des Problems.

Theorem 3 *Jede Durchschnittsgeschwindigkeit* $\frac{||(\delta x,\delta y)||}{\delta t}$ *eines kürzesten Pfades ist konstant und entspricht der maximalen Robotergeschwindigkeit* V*.*

Beweis. Wir nehmen an, dass ein gegebener Pfad π mindestens ein Teilstück π_1 besitzt, welches mit einer Geschwindigkeit $v_{\pi_1} < V$ zurückgelegt wird. Dies kann dann jedoch kein kürzester Pfad sein, da π_1 auch mit der maximalen Geschwindigkeit V hätte zurückgelegt werden können, um das Ziel in früherer Zeit zu erreichen. Der Pfad mit schnellerem π_1 ist zudem garantiert auch kollisionsfrei, wie in Lemma 2 gezeigt wurde. □

3.3 Teilpfade von kürzesten Pfaden

Nachdem wir gezeigt haben, welche Bedeutung Zeit und Geschwindigkeit für kürzeste Pfade haben, widmen wir uns nun der Betrachtung von Teilstücken kürzester Pfade.
Dabei werden wir sehen, dass kürzeste Pfade stets aus Abschnitten von geradlinigen Segmenten und Kurven, welche sich an die Hindernis-Kegel schmiegen, bestehen.

Theorem 4 *Ein kürzester Pfad besteht abschnittsweise aus geradlinigen Stücken und Abschnitten, die auf Kegelrändern liegen.*

Beweis. Innerhalb des freien Raumes ist die kürzeste Verbindung zwischen zwei Punkten eine gerade Linie. Falls ein Hindernis im Weg liegt, so wird dieses so nah wie möglich umfahren, um wieder auf eine gerade Linie zurückzukehren. In unserem Fall sind die Hindernisse Kegel, auf deren Oberfläche nahes Umfahren in Kurven resultiert. □

3.4 Kurven auf Kegeln

In diesem Abschnitt wollen wir die Gestalt der Kurven näher betrachten, welche beim Umfahren der Kegel entstehen. Wir werden dabei sehen, dass es sich um logarithmische Spiralen handelt, auf die wir dann näher eingehen.

Annahme: Der zu betrachtende Kegel hat seine Spitze der Einfachheit halber im Koordinatenursprung, die korrespondierende Scheibe wächst mit der konstanten für dieses Hindernis festgelegten maximalen Geschwindigkeit $v_i = 1$ und hat zu Beginn $t = 0$ den Radius $r = 0$. Diese Annahme ist zulässig, weil sich andere Parametrisierungen nicht auf die im Folgenden hergeleiteten Formeln auswirken.
Die Koordinaten drücken wir als Polarkoordinaten $(r(t), \theta(t))$ zum Zeitpunkt t aus. $r(t)$ entspricht dann trivialerweise durch das konstante Wachstum des Kegels der Zeit, also:

$$r(t) = t$$

Den Winkel können wir leider nicht so einfach ableiten. Jedoch lässt sich eine geschlossene Form herleiten.
Da die Pfadgeschwindigkeit konstant V ist, können wir daraus Folgerungen ziehen:

$$\sqrt{x'(t)^2 + y'(t)^2} = V$$

Nun wird

$$\{x(t) = r(t)\cos\theta(t), y(t) = r(t)\sin\theta(t)\}$$

abgeleitet zu:

$$\{x'(t) = r'(t)\cos\theta(t) - r(t)\theta'(t)\sin\theta(t), y'(t) = r'(t)\sin\theta(t) + r(t)\theta'(t)\cos\theta(t)\}$$

Die Quadrate davon bilden:

$$x'(t)^2 = r'(t)^2\cos^2\theta(t) - 2r'(t)r(t)\theta'(t)\sin\theta(t)\cos\theta(t) + r(t)^2\theta'(t)^2\sin^2\theta(t)$$

$$y'(t)^2 = r'(t)^2\sin^2\theta(t) + 2r'(t)r(t)\theta'(t)\sin\theta(t)\cos\theta(t) + r(t)^2\theta'(t)^2\cos^2\theta(t)$$

aufsummiert fallen die gemischten Terme weg:

$$x'(t)^2 + y'(t)^2 = r'(t)^2\cos^2\theta(t) + r'(t)^2\sin^2\theta(t) + r(t)^2\theta'(t)^2\sin^2\theta(t) + r(t)^2\theta'(t)^2\cos^2\theta(t)$$

und mit Ausklammern kommen wir zu:

$$x'(t)^2 + y'(t)^2 = r'(t)^2(\sin^2\theta(t) + \cos^2\theta(t)) + r(t)^2\theta'(t)^2((\sin^2\theta(t) + \cos^2\theta(t))$$

Weil $\sin^2\theta(t) + \cos^2\theta(t) = 1$ folgt:

$$x'(t)^2 + y'(t)^2 = r'(t)^2 + r(t)^2\theta'(t)^2 = 1 + t^2\theta'(t)^2$$

und schließlich eingesetzt:

$$\sqrt{1 + t^2\theta'(t)^2} = V$$

$$1 + t^2\theta'(t)^2 = V^2$$

aufgelöst nach $\theta'(t)$:

$$\theta'(t) = \pm\frac{\sqrt{V^2 - 1}}{t}$$

und wieder integriert:

$$\theta(t) = \pm\sqrt{V^2 - 1}\log t + \theta_0$$

Diese letzte Gleichung für $\theta(t)$ genügt der Winkelgleichung logarithmischer Spiralen:

Logarithmische Spirale [Weisstein07] in Polarkoordinaten:

$$r = ae^{b\theta}$$

nach θ aufgelöst:

$$\theta = \frac{1}{b}\ln(\frac{r}{a})$$

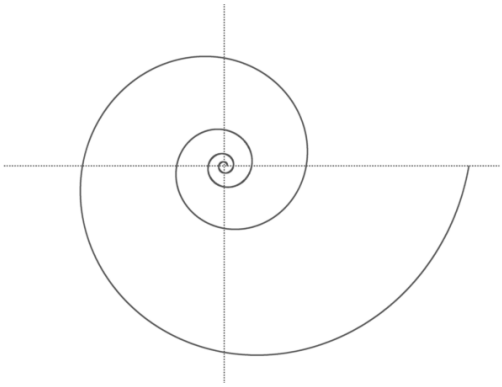

Abbildung 3: Mit jeder Umdrehung weitet sich eine logarithmische Spirale um einen konstanten Faktor. Darin drücken sich die zunehmend längeren Wege bei einem Umlauf auf dem wachsenden Kegelrand anschaulich aus.

Da r zu t korrespondiert ist die Ähnlichkeit der Formeln ersichtlich. Das Vorzeichen von $\theta(t)$ indiziert, ob es sich um eine Drehung im Uhrzeigersinn (-) oder gegen den Uhrzeigersinn (+) handelt, θ_0 gibt den Anfangswinkel an.

3.5 Glattheit des Pfades

Theorem 5 *Der kürzeste Pfad hat keine scharfen Ecken, ist somit C^1-glatt.*

Beweis. Sei der Pfad π ein kürzester Pfad und nicht glatt, habe also mindestens eine Ecke. Dann könnte man diese Ecke durch eine direkte Verbindung oder ein Spiralstück abkürzen. Daher kann π kein kürzester Pfad sein. □

Dies führt zu dem Schluss, dass der ganze Pfad aus glatten Strecken besteht, abwechselnd aus geradlinigen Elementen und Spiralen um die Kegel. Letztere Spiralen werden durch bitangente Strecken zwischen den Kegeln verbunden.

3.6 Absprungkurven

Wie gerade festgestellt, befinden sich zwischen den Kegeln bitangente Strecken, die diese verbinden. Insgesamt gibt es vier Möglichkeiten, wie diese bitangenten Strecken verlaufen können, links-links, links-rechts, rechts-links und rechts-rechts an den Seiten des Kegel vorbei. Da die Tangenten in verschiedenen Winkeln verlaufen können, gibt es unendlich viele Möglichkeiten, wie diese an beiden Kegeln anliegen. Die Berührpunkte bilden dabei geschlossene Kurven auf den beiden Kegeln, welche wir im folgenden als Absprungkurven bezeichnen. Uns interessieren aber insbesondere die Kurven, deren Tangenten eine Steigung von $\frac{1}{V}$ haben, somit also maximale Geschwindigkeit.

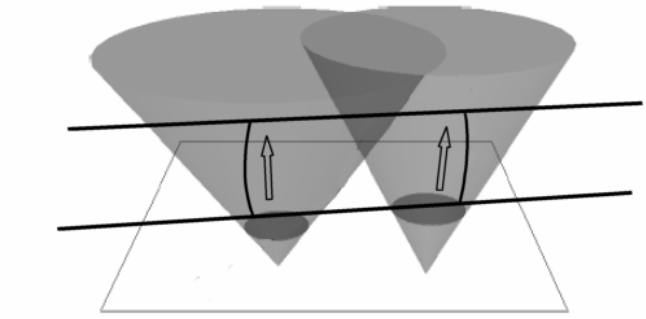

Abbildung 4: Wird eine Tangente konstanter Steigung an der Kegeloberfläche entlang geschoben, bilden sich kontinuierliche Kurven.

Definition 6 Die Menge der Punkte, welche zwei Kegel C_i, C_j mit der Steigung $\frac{1}{V}$ bitangent verbinden, wird mit DC(C_i, C_j) bezeichnet und enthalten vier Kurven, die die zuvor genannten Fälle der Orientierung repräsentieren. Die Verbindung von einem Kegel $C_i$zu dem Zielpunkt g wird mit DC(C_i,g)

bezeichnet und besteht nur aus zwei Kurven, die sich auf der linken und rechten Seite des Kegels befinden.

Im Folgenden betrachten wir die Parametrisierung von Punkten auf der Oberfläche von

$$C : (T, \Theta) \rightarrow \{T\cos\Theta, T\sin\Theta, T\}.$$

Dies entspricht den euklidischen Koordinaten. x- und y-Koordinate sind durch sin und cos mit gleichem Winkel Θ ausgedrückt, was einem Punkt auf dem Einheitskreis entspricht. Dieser wird durch den Faktor T der Zeit entsprechend nach außen verschoben. Da die Kurven alle die gleiche Steigung haben, gibt es zu jedem Punkt auf der Kegeloberfläche nur eine korrespondierende Spirale, die durch diesen Punkt verläuft, welche durch den Anfangswinkel θ_0 definiert wird:

$$\theta_0 = -\sqrt{V^2 - 1}\log T + \Theta$$

Betrachten wir nun also eine rechtsdrehende Spirale und setzen in die zuvor hergeleitete Gleichung

$$\theta(t) = \pm\sqrt{V^2 - 1}\log t + \theta_0$$

den Anfangwinkel ein, so erhalten wir für die x- und y-Koordinaten unserer Spirale:

$$\{x(t) = t\cos(\sqrt{V^2 - 1}\log t - \sqrt{V^2 - 1}\log T + \Theta),$$
$$y(t) = t\sin(\sqrt{V^2 - 1}\log t - \sqrt{V^2 - 1}\log T + \Theta)\}$$

Daraus können wir eine Gleichung für eine Tangente ableiten:

$$l(t) = \{x(T) + (t - T)x'(T), y(T) + (t - T)y'(T)\}$$

In die wir die Koordinaten der Spirale einsetzen:

$$= \{t\cos\Theta - (t - T)\sqrt{V^2 - 1}\sin\Theta, t\sin\Theta + (t - T)\sqrt{V^2 - 1}\cos\Theta\}$$

Man kann gut erkennen, dass eine Wahl $t = T$ dazu führt, dass sich die zuvor gewählte Punktdarstellung der Spirale in euklidischen Koordinaten ergibt.

Nun wollen wir ermitteln, wie und ob diese Tangente den anderen Kegel schneidet. Damit das hergeleitete Linienstück gleichzeitig tangent zu beiden Kegeln ist, somit zu einem Kurvensatz DC gehört, setzen wir die beiden Koordinatenanteile in die Kegelgleichung des „Zielkegels“ ein.

$$C_i : (x - p_{ix})^2 + (y - p_{iy})^2 = (v_i t + r_i)^2$$

Lösen wir dies nach t, dem Ankunftszeitpunkt, auf:

$$t_{1,2} = A(T, \Theta) \pm \sqrt{D(T, \Theta)}.$$

Dies liefert uns den Term der Ankunftszeit $A(T, \Theta)$, wobei der zweite Teil, $\sqrt{D(T, \Theta)}$ beschreibt, ob die Linie den zweiten Kegel schneidet, berührt (der für uns interessante Fall, $D(T, \Theta) = 0$) oder verfehlt.

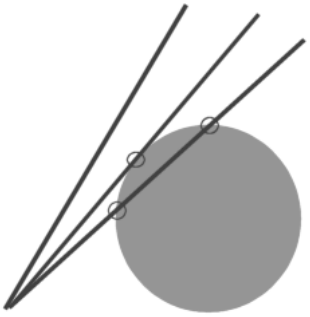

Abbildung 5: Je nach Ergebnis des Wurzelterms schneidet die Tangente den Zielkegel zweimal ($D(T,\Theta) > 0$), tangiert ihn wie gewünscht ($D(T,\Theta) = 0$) oder verfehlt ihn vollständig ($D(T,\Theta) < 0$).

4 Ein naiver Algorithmus

4.1 Idee und Initialisierung

Ausgehend von den bisher aufgezeigten Überlegungen sind wir im Stande nun einen ersten Algorithmus anzugeben, der einen kürzesten Pfad finden kann.
Wir werden dazu einen Suchbaum aufbauen, der die verschiedenen, möglichen Pfade entlang der Hindernisse repräsentiert.
Die Wurzel des Baums liegt bei der Zeitkoordinate 0 und in unserem Startpunkt s. In einer Priority Queue verwalten wir die Blätter und expandieren jeweils dasjenige zuerst, welches die aktuell kleinste Zeitkoordinate hat, denn diese entspricht auch der Länge des Pfades, da der Roboter sich, wie in Theorem 3 gezeigt, immer mit konstanter, maximaler Geschwindigkeit V bewegt.
Die Initialisierung der Queue erfolgt entweder mit einer geradlinigen Verbindung, die sofort zum Ziel führt, oder mit einer Tangente zu den Kegeln, die den Weg versperren, die Steigung dieser Linien beträgt $\frac{1}{V}$. Falls eines der Segmente auf dem Weg bereits einen Kegel schneidet, so ist es ungültig und kann entfernt werden.

4.2 Ablauf des Algorithmus

Algorithmus 1: ShortestPathNaive(s,g)
1: Priority Queue Q mit Endpunkten aller gültiger ausgehender Linien initialisieren.
2: while Q ist nicht leer do
3: Frontelement (q,t) aus der Queue nehmen.
4: if g ist zu Zeitpunkt t nicht kollisionsfrei then
5: Es existiert kein Pfad, terminieren.
6: else if q = g then
7: Kürzester Pfad gefunden, terminieren.
8: else
9: q liegt auf der Oberfläche eines Kegel Q_i, mit Spirale um
10: C_i laufen bis Kollision oder eine Absprungkurve gefunden wird
11: if Spirale trifft Absprungkurve DC(C_i,C_j) then
12: (q',t') als Schnitt von DC und Spirale berechnen
13: (q",t") Ankunftspunkt der Bitangente auf C_j berechnen
14: (q',t') in Q einfügen
15: if Verbindung (q',t'),(q",t") kollisionsfrei then
16: (q",t") in Q einfügen
17: Es existiert kein kürzester Pfad.

Nach der Initialisierung und jedem Berechnungsschritt wird das erste Element (jenes mit dem kleinsten t-Wert und somit den besten Aussichten, ein kürzester Pfad zu sein) aus der Priority Queue entnommen. Hierbei sind zwei Fälle zu unterscheiden:

1. Das Element ist die Zielkonfiguration. In diesem Fall ist direkt ein kürzester Pfad gefunden, denn die Zeitkoordinate dieses Punktes ist minimal gegenüber allen Weiteren, die sich noch in der Quere befinden.

2. Das Element liegt auf einem der Kegel in einem Berührpunkt. Gemäß der Vorüberlegungen wird nun auf einer Spirale der Kegel umlaufen, bis ein neuer Absprungpunkt gefunden wird oder ein zweiter Kegel den Weg schneidet und somit für ungültig erklärt.

Wird ein Absprungpunkt auf einer der Absprungkurven gefunden, so verästelt sich der Suchbaum folgendermaßen:
Ein Ast folgt der Tangente und trifft auf einem weiteren Kegel auf. Falls diese Tangente keinen fremden Kegel schneidet, wird der Auftreffpunkt in die Queue übernommen. Ein zweiter Ast folgt der Spirale auf dem ursprünglichen Kegel, bis sie eine weitere Absprungkurve schneidet, oder auf ein Hindernis trifft (Schnitt mit einem fremden Kegel).

Bei den oberen Schnitten und Spiralläufen bleibt zu klären, welche Orientierungen dabei zum Zuge kommen. Diese lassen sich jedoch mit den zuvor erarbeiteten Formeln ermitteln:

Angenommen wir haben einen Punkt q auf einem Kegel erreicht und wollen nun die Spirale identifizieren, die dazu korrespondiert. Nach den Aussagen des vorigen Kapitels gibt es nur eine Spirale, die durch diesen Punkt verlaufen kann und durch einen berechenbaren Startwinkel θ_0 beschrieben wird. Bei gegebenen euklidischen Koordinaten (x, y, t) ergeben sich die Polarkoordinaten $(T, \Theta) = (t, \arctan \frac{y-p_{ix}}{x-p_{ix}})$.
Wie schon zuvor gezeigt, lässt sich aus dieser Darstellung der Anfangswinkel, wie auch der Winkel zu einem beliebigen Zeitpunkt t ableiten:

$$\theta_0 = -\sqrt{V^2 - 1} \log T + \Theta$$

$$\theta(t) = \pm\sqrt{V^2 - 1} \log t + \theta_0$$

Desweiteren interessieren uns die Schnittpunkte von Spiralen und Absprungkurven DC. Da wir bereits eine explizite Gleichung für die Absprungkurve hergeleitet haben, müssen wir unser Ergebnis für den Schnitt lediglich einsetzen und nach t auflösen:

$$D(t, \theta(t)) = 0$$

Nach dem Bestimmen der Zeit t muss noch ermittelt werden, ob die Tangente zu dem nächsten Kegel auf der linken oder rechten Seite ankommt. Dies ist insbesondere wichtig, um festzulegen, ob der Kegel von einer Spirale im Uhrzeigersinn (links ankommend) oder gegen den Uhrzeigersinn (rechts ankommend) umkreist wird.
Dies lässt sich durch die Ableitung von $D(t, \theta(t))$ ermitteln. Dabei entspricht ein negativer Wert links und ein positiver rechts. Man vergleiche zur besseren Vorstellung dazu Abb. 5. Eine negative Ableitung reduziert den Term unter der Wurzel weiter, wodurch die Wurzel keine reellwertigen Lösungen mehr hat, dies impliziert Ankunft links. Eine positive Ableitung erhöht den Term unter der Wurzel, somit ergeben sich dann zwei Lösungen, dies impliziert Ankunft rechts.
Der Ankunftszeitpunkt ergibt sich aus dem Teil $A(T, \theta(T))$. Falls der Zeitpunkt kleiner T ist (in diesem Fall verläuft die Tangente „nach unten“), muss sie nicht betrachtet werden, weil sie ungültig ist. Ansonsten ergibt sich dadurch die genaue Ankunftsposition auf dem zweiten Kegel, berechnet durch die Tangentengleichung: $l(A(T, \theta(T)))$.

4.3 Ergebnis

Der Algorithmus endet erfolgreich, falls das Ziel g als Frontelement der Queue entnommen wird.

Falls die Queue leer ist, also der Baum vollständig in ungültigen Blättern geendet ist oder aber die Zielposition zum Zeitpunkt des ersten Elements bereits innerhalb eines Kegels liegt, existiert kein gültiger, kürzester Weg von s nach g.

5 Eine effiziente Implementierung

5.1 Idee

Der oben vorgestellte naive Algorithmus wird zwar eine Lösung in endlicher Zeit finden, falls es eine solche gibt, jedoch kann es passieren, dass der dabei erstellte Baum in seiner Komplexität explodiert. Aus diesem Grund bietet es sich an, nach Knoten zu suchen, die nur ein einziges Mal besucht werden, so dass man eine Kürzeste-Wege Suche nach Dijkstra nutzen kann. Dabei unterscheiden wir den Fall gleicher Wachstumsrate und verschiedener Wachstumsrate der einzelnen Kreisscheiben.

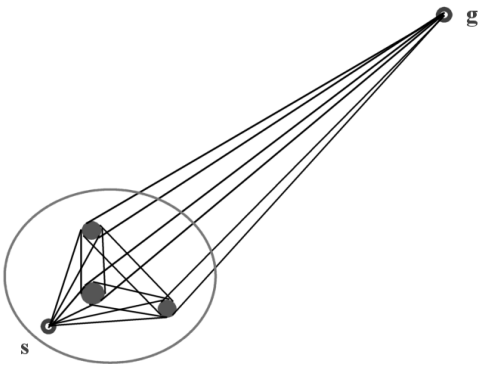

Abbildung 6: Bei ungünstiger Konfiguration werden in dem hier mit einem Kreis markierten Teil sehr viele „mögliche“ kürzeste Wegstücke erzeugt, die die Queue extrem aufblähen und zu keinem kürzesten Pfad mehr führen können, sobald die Kegel bereits vollständig umkreist wurden. Der naive Algorithmus berücksichtigt jedoch trotzdem alle dieser Punkte, da die Zielkonfiguration g erst zu einem späteren Zeitpunkt entnommen wird und dann erst der Ablauf terminiert wird.

5.2 Gleiche Wachstumsrate der Kreisscheiben

Mit Hinblick auf Theorem 3 können wir über einen Punkt $q = (T, \Theta)$ auf einer Kegeloberfläche sagen, dass alle Punkte q', die von diesem Punkt aus mit Geschwindigkeit $V' < V$ erreichbar sind, nicht auf einem kürzesten Pfad liegen können. Daher beschränken die beiden Spiralen im und gegen

den Uhrzeigersinn durch den Punkt q eine Fläche uninteressanter Punkte, die wir im Folgenden Keilfläche (engl. Wedge Region) von q nennen.

Leiten wir nun daraus Knoten ab. Sei uns die Darstellung der Kegeloberfläche bis zu dem Zeitpunkt durch Berechnung gegeben, bei der die Zielkonfiguration ungültig, also erstmalig von einem beliebigen Kegel geschnitten wird. Die Oberfläche enthalte alle Absprungkurven und Hindernisse, d.h. Schnitte mit anderen Kegeln. (vgl. Abb. 1), ebenso seien die Stücke der Absprungkurven berechnet, deren ausgehende Tangenten fremde Kegel schneiden.

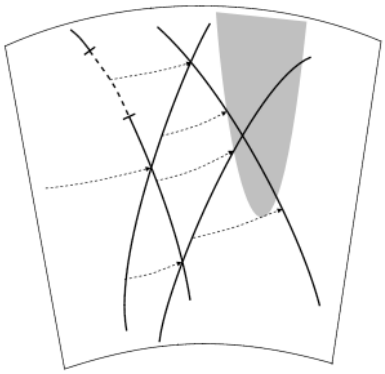

Abbildung 7: Eine mögliche Kegeloberfläche. Schwarze Linien entsprechen Absprungkurven. Die dünnen Linien begrenzen Intervalle auf diesen Kurven. Das dick gestrichelte Stück ist ein Schattenintervall, worauf später eingegangen wird. Die graue Fläche ist der Schnitt mit einem anderen Kegel.

In dem von uns betrachteten Fall haben alle Scheiben die gleiche Wachstumsrate, was uns folgende Eigenschaft liefert:
Wenn $q_1 = (T, \Theta)$ ein Punkt auf einer Absprungkurve ist, dann sind alle Punkte (T', Θ') mit $T' > T$ innerhalb der aufgespannten Keilfläche von q_1. Der Beweis hierfür sei ausgelassen, da zu zeigen ist, dass eine Bewegung direkt entlang der Absprungkurven zu anderen Kegeln mit Geschwindigkeit $\frac{||(\delta x, \delta y)||}{\delta t} \leq V$ geschieht. Intuitiv kann man sich jedoch schon vorstellen, dass diese Kette von Tangentenberührpunkten auf der Kegeloberfläche recht steil verläuft, wie man auch in der Grafik angedeutet erkennen kann.
Aus dieser Tatsache können wir schließen, dass die Intervalle der Absprungkurven direkt als Knoten dienen können, da sie Zwischenpunkte kürzester Wege sind. Wenn zu späterem Zeitpunkt ein weiterer Pfad in diesem Intervall ankommt, dann hätte dieser Punkt bereits zuvor erreicht werden können, indem von dem ersten aus entlang der Absprungkurve auf dem Kegel gefolgt wird.

Wie auch schon im naiven Algorithmus gehen nun von jedem Knoten zwei Kanten aus. Die erste Kante entspricht dem Spiralstück zum nächsten Schnittpunkt mit einer anderen Absprungkurve auf dem gleichen Kegel und die zweite Kante entspricht einer Tangente mit anhängendem Spiralstück zu der ersten Absprungkurve des nächsten Kegels.

Um nun Positionen auf den Kegeloberflächen, oder auch den nächsten Schnittpunkt einer Spirale mit einer Absprungkurve effizient ermitteln zu können, bedienen wir uns der Datenstruktur einer Trapezkarte, wobei die Seiten der Trapeze den Spiralstücken entsprechen.

5.3 Laufzeit

Theorem 7 *Der vorgestellte Algorithmus zur Berechnung eines kürzesten Pfades bei Scheiben gleicher Wachstumrate hat die Laufzeit* $O(n^3 \log n)$.

Beweis. Jedes Paar von Kegeln kann nur $O(1)$ Absprungkurven haben (genau 4). Da wir n Kegel haben und alle mit allen paaren müssen, ergeben sich somit $O(n^2)$ Absprungkurven. Diese wiederum können in maximal $O(n)$ Intervalle zerfallen, da sie nur von maximal n Kegeln geschnitten werden können und die Intervalle in höchstens zwei Teile trennen. Somit gibt es insgesamt $O(n^3)$ Teilintervalle mit $O(1)$ ausgehenden Kanten.
Wir wissen, dass der Dijkstra Algorithmus die Laufzeit $O(N \log N + E)$ hat, wobei N die Anzahl der Knoten und E die Anzahl der Kanten bedeutet.
An zusätzlichen Kosten müssen wir einerseits für jede Kante ermitteln, welches das nächste Ankunftsintervall auf einer Absprungkurve ist und im Falle einer Tangente, ob diese kollisionsfrei zum nächsten Kegel führt. Beides kann in $O(\log n)$ erledigt werden, da die Trapezkarte zum einen logarithmische Suchzeit bietet und die Schattenintervalle von ungültigen Absprungstellen beim Aufbau der Trapezkarten mitberechnet werden können.
Die Berechnung der Trapezkarten benötigt $O(n^2)$ für jeden Kegel, was sich bei n Kegeln zu $O(n^3)$ ergibt. Die Schattenstellen auf den Absprungkurven lassen sich auch in $O(n^3)$ berechnen, wenn die $O(n^2)$ Absprungkurven mit allen $O(n)$ Kegeln geschnitten werden.

Dies führt zu einer Gesamtlaufzeit von $O(n^3 \log n)$. □

5.4 Unterschiedliche Wachstumsrate der Kreisscheiben

Während sich der vorige Fall gut behandeln ließ, finden sich bei verschiedenen Wachstumsraten Probleme, die sich nicht ohne Weiteres lösen lassen.
Die Berechnung der Absprungkurven wird deutlich komplizierter. Wenn für eine Kurve $DC(C_i, C_j)$ der erste Kegel C_i schneller wächst als C_j, dann wird

auch die Geschwindigkeit auf der Kurve noch kleiner als der maximalen Geschwindigkeit V des Roboters sein.
Jedoch kann der umgekehrte Fall dazu führen, dass die Absprungkurve in manchen Punkten sehr flach oder sogar horizontal verläuft, was eine unendlich große Geschwindigkeit voraussetzt. Somit können in diesem Fall keine Intervalle auf den Kurven mehr berechnet und unterschieden werden.
Außerdem können sich noch deutlich mehr Verzweigungen bilden, da zum Beispiel um eine sehr langsam wachsende Scheibe beliebig viele Umläufe durchgeführt werden können, die alle zu weiteren Wegen führen.

5.5 Konkrete Anwendung

Um den Suchbaum klein zu halten ist intensives Pruning notwendig. Dazu verwenden wir eine Beobachtung, die wir schon zuvor gemacht haben, nämlich bezüglich der Keilflächen auf den Kegeloberflächen.
Sei (T, Θ) ein Punkt auf einem Kegel. Dieser kann genau dann nicht mehr zu einem kürzesten Pfad gehören, wenn der gleiche Winkel Θ schon früher in einem Punkt (T', Θ) mit $T' < T$ besucht wurde und die vertikale Verbindung von T' und T kollisionsfrei ist.
Für die praktische Umsetzung wird nur eine konstante Nummer an Tests für Winkel Θ auf den Kegeln ausgeführt, die gemäß $\frac{2\pi}{\epsilon}$ gleichverteilt sind. ϵ stellt dabei die Anzahl der vertikalen Linien auf der Kegeloberfläche dar. Diese Linien werden von den anderen Kegeln geschnitten und in kollisionsfreie Regionen unterteilt, somit können wir nun diese als Knoten für den Dijsktra Algorithmus benutzen. Jedoch lässt sich die Laufzeit nun nicht mehr in der Größenordnung von Hindernisscheiben angeben.

5.6 Details der direkten Implementierung

Im Folgenden möchte ich noch auf die Implementierung von Jur van den Berg und Mark Overmars eingehen. Um die Geschwindigkeit des Algorithmus bei verschiedenen Wachstumsraten der Hindernisse zu ermitteln, programmierten sie eine Anwendung, die den Vorgang der Pfadsuche ausführt und visualisiert.
Die Pfadsuche geschieht mit dem um Pruning erweiterten Algorithmus, wobei nicht sämtliche Vertikalen aller Kegel berechnet werden, sondern für die korrespondierenden Winkel nur ein Zeitwert gespeichert wird, wann die Stelle zuletzt besucht wurde.
Falls ein Punkt q in einem Zeitpunkt q_t betrachtet wird, wird geprüft, ob die entsprechende Vertikale schon einmal zu einem früheren Zeitpunkt t_m überschritten wurde und ob die direkte Verbindung auf der Kegeloberfläche kollisionsfrei ist. Falls ja, dann befindet sich q im gleichen Intervall und kann ignoriert werden, falls nein, dann werden die ausgehenden Kanten wieder in

die Queue sortiert und der Zeitwert wird auf q_t gesetzt.
Desweiteren werden nicht mehr aufwändige Spiralstücke zwischen benachbarten Vertikalen berechnet, sondern stattdessen einzelne gerade Strecken, was zwar Unsicherheit einbringt, aber auf Grund der Wahl von ϵ vernachlässigbar sei.
Auch wurde der Dijkstra Algorithmus gegen eine A* Suche ausgetaucht, welche zielgerichteter ist und in der Praxis bessere Laufzeiten erzielen kann. Die untere Grenze der verwendeten Heuristik stellt dabei die direkte euklidische Entfernung zum Ziel dividiert durch die maximale Geschwindigkeit V dar.

5.7 Ergebnisse der Experimente

Die Experimente wurden auf einem Pentium IV 3.0GHz mit 1 GByte Arbeitsspeicher ausgeführt. Um verlässliche Werte zu erhalten, wurden für eine Anzahl n an Scheiben mehrere Anfangskonfigurationen mit Anordnung von Start/Ziel und der Scheiben ausprobiert.
Dabei erwies sich die Implementierung als sehr schnell, da selbst bei 15 Scheiben nur 0.0042 Sekunden benötigt wurden. Insgesamt scheint sich die Zahl der Scheiben ungefähr quadratisch auf die Laufzeit auszuwirken, wie man an der Grafik ungefähr schätzen kann.

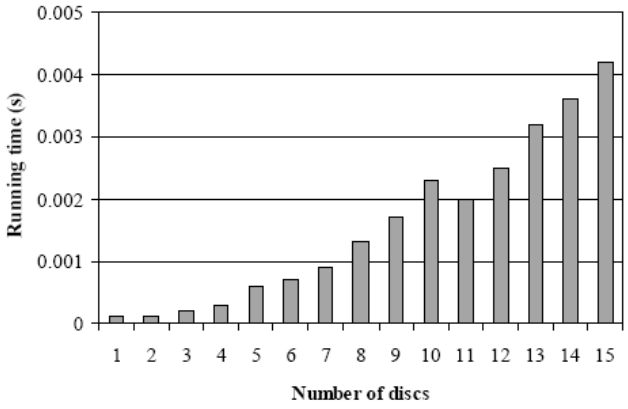

Abbildung 8: Anzahl der Scheiben und gemessene Laufzeiten der Applikation von van den Berg und Overmars. Die Unregelmäßigkeit bei 11 Scheiben ergibt sich durch zufällige Generierung der Umgebungen.

Literatur

[Berg06] Van den Berg, J., Overmars, M. *Planning the Shortest Safe Path amidst Unpredictably Moving Obstacles*. 2006.

[Weisstein07] Weisstein, E. W. *The Logarithmic Spiral*. 2007. Veröffentlicht unter: http://mathworld.wolfram.com/LogarithmicSpiral.html