

Kapitel IV: Bewegungsplanung i.d. Ebene.

4.1. Einführung:

4.1.1. Allgemeines Problem:

Geg: Eine Szene S von Hindernissen (Polygone, Segmente, ...), ein Roboter R (Polygon, Kreisscheibe, ...) und zwei Pkte A, B .

Aufgabe: Beantworte die Frage, ob R von A nach B bewegt werden kann, ohne mit Hindernissen aus S zu kollidieren bzw. gib einen kollisionsfreien Weg an.

→ Piano Moves Problem.

4.1.2. Bemerkungen:

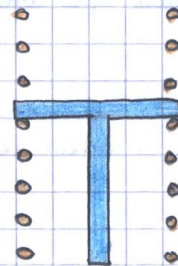
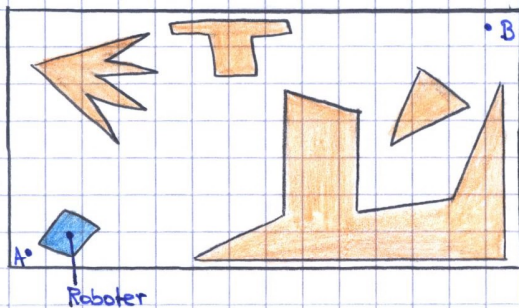
i) Wir behandeln zwei Spezialfälle:

- S ist Menge von Kurvensegmenten u. R ist eine Kreisscheibe.
- S ist Menge von konvexen Polygone, R ist konvexes Polygon.

ii) Bewegung: nur Translationen.

Es gibt Vielzahl weiterer Varianten:

- Rotationen erlaubt
- Kurzester Weg.
- Sicherster Weg.



4.2. Problem 1: R ist Kreis und S Menge von Segmenten.

4.2.1. Idee: Versuche den sichersten Weg zu finden, d.h. R hält immer max. möglichen Abstand zu den Segmenten.

4.2.2. Frage: Was sind die Orte mit maximalen Abstand zu allen Hindernissen?

4.2.3. Antwort: Alle Pkte auf (Knoten &) Kanten des Voronoi-Diagramms von S . V(DS)

4.2.4. Voronoi-Diagramm von Segmenten i.d. Praxis:

→ annäherung durch setzen von Hilfpunkten (genügend dicht) und Berechnung des VD's dieser Pkte.

Anschließend Erzeugung aller Voronoi-Kanten, die von Segmenten geschnitten werden.

4.2.5. Definition: $\forall p \in \mathbb{R}^2$ ist

a) Freiheit (p): = minimaler Abstand von p zu allen Hindernissen.

b) p heißt frei \Leftrightarrow Freiheit(p) $\geq r$, wobei r = Radius von R .

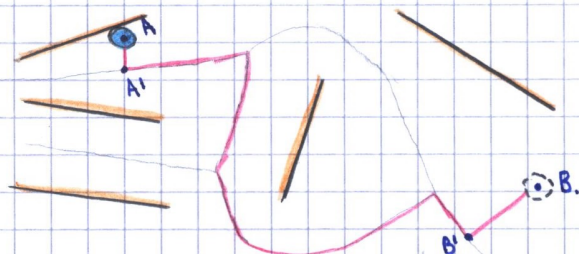
(d.h. Kreisscheibe R kann mit Mittelpunkt auf p platziert werden, ohne ein Segment zu schneiden).

c) FP := $\{p \in \mathbb{R}^2; p \text{ ist frei}\}$

4.2.6. Idee für Algorithmus:

- Bewege R von Anfangsposition A (Ann. freie Position) auf nächste Voronoi-Kante (\rightarrow Position A').
- Bewege R auf Voronoi-Kanten, die mindestens Freiheit r haben, so nah wie möglich an B heran. (\rightarrow Position B')
- Bewege R von B' nach B (Ann. B ist frei).

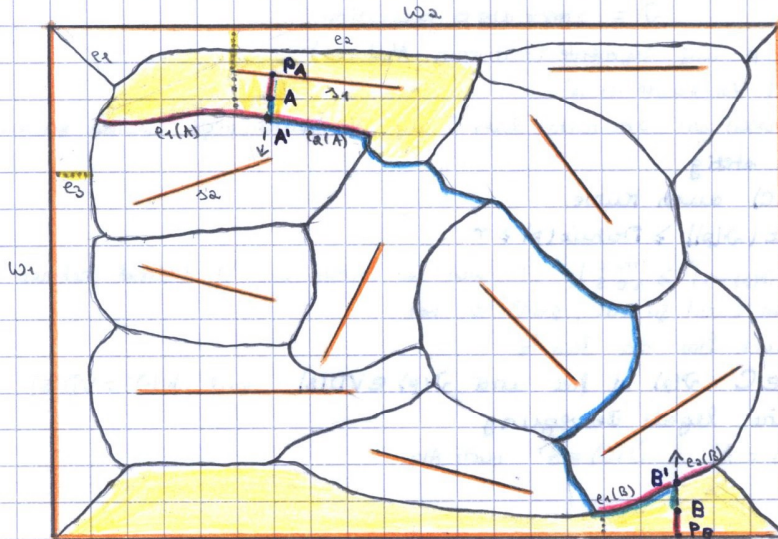
4.2.7. Beispiel:



4.2.8 Algorithmus:

1. Konstruiere $VD(S)$. Speichere für jede Kante die beiden Orte, die e definieren.
(entweder Segmentendpunkte oder Segmentwand).
Und $\text{Freiheit}(e) :=$ minimaler Abstand eines PKtes auf e zu einem Segment.
($\text{Freiheit}(e) \geq r \Rightarrow R$ kann über e bewegt werden).
↑ Preprocessing.
2. Bestimme für A und B jeweils den nächsten Pkt P_A bzw. P_B auf einem Segment durch lineare Suche auf S (\rightarrow Point Location)
(P_A, P_B sind Schnittpkte des Lots von A bzw. B auf nächstes Segment).
D.h. Betrachte Lot von A bzw. B auf alle Segmente in S , suche nach einem Lot, der eine Voronoi-Kante schneidet \rightarrow lin. Suche.
3. $A' \leftarrow$ erster Schnittpkt des Strahls $\overrightarrow{P_A A}$ mit $VD(S)$
 $B' \leftarrow$ erster Schnittpkt des Strahls $\overrightarrow{P_B B}$ mit $VD(S)$
Falls A' bzw. B' keine Voronoi-Knoten, dann mache sie zu künstlichen Knoten durch spalten der entsprechenden Kante e in e_1, e_2 .
 \rightarrow Berechne $\text{Freiheit}(e_1), \text{Freiheit}(e_2)$!
4. Suche einen Pfad P in $VD(S)$ von A' nach B' mit $\text{Freiheit}(e) \geq r \forall e \in P$.
 \approx B. durch DFS oder BFS.
Falls kein solcher Pfad existiert, melde "keine Lösung".
 \rightarrow STOP.
5. Der Weg $A \rightarrow P \rightarrow B$ ist eine mögliche Bewegung.

4.2.9 Bsp:



1. für e_1 speichere: (w_1, w_2) $\text{Freiheit}(e_1) = 0 \text{ cm}$
für e_2 speichere: (s_1, w_2) $\text{Freiheit}(e_2) = 0,5 \text{ cm}$
für e_3 speichere: (Endpkt von s_2, w_1) $\text{Freiheit}(e_3) = 0,5 \text{ cm}$ u.s.w. ...
2. Berechne Lots von A und B auf alle Segmente (incl. Ränder)
Von den berechneten suche die mit dem kleinsten Abstand aus (lin. Suche)
Dadurch wissen wir in welcher Region sich Ort A und B befinden (point location)
Im Bsp. sind die jeweiligen Regionen s_1, s_2 .
3. Teile $e_1(A)$ und $e_1(B)$ und $e_2(B)$ in $e_1(B)$ und $e_2(B)$
 $\text{Freiheit}(e_1(A)) = 0,6 \text{ cm}$, $\text{Freiheit}(e_2(A)) = 0,7 \text{ cm}$
 $\text{Freiheit}(e_1(B)) = 0,3 \text{ cm}$, $\text{Freiheit}(e_2(B)) = 1 \text{ cm}$.
4. $\forall e \in \text{blau}$ gilt: $\text{Freiheit}(e) \geq r$

4.2.10 Lemma:

- 1) falls A bzw. B frei ist $\Rightarrow \exists$ Bewegung von A nach A' bzw. von B nach B'.
- 2) \exists Bewegung von A nach B $\Leftrightarrow \exists$ Bewegung von A' nach B', die nur Voronoi-Kanten besitzt.

Beweis:

1) Sei $VR(S)$ die Voronoi-Region, die A enthält

$$\Rightarrow P_A \in S \cup \{S\}$$

Seien A, B frei \Rightarrow Freiheit(A) $\geq r \wedge$ Freiheit(B) $\geq r$

$$\text{Freiheit}(A) := \min_{s \in S} |A-s|, \quad \text{Freiheit}(B) := \min_{s \in S} |B-s|$$

Es gilt: $r \leq \text{Freiheit}(A) \leq \text{Freiheit}(p) \forall p \in \overline{AA'}$

\Rightarrow Es gibt eine legale Bewegung von R von A nach A'

weil Freiheit(p) $\geq r \forall p \in \overline{AA'}$.

$B \rightarrow B'$ analog.

2) " \Leftarrow ": \exists Bewegung von A' nach B', die nur Voronoi-Kanten besitzt.

Beh folgt nach Teil 1), falls A und B frei sind.

" \Rightarrow ": z.z. \exists Bew. von A' nach B', die nur Vor. K. besitzt.

Eine beliebige legale Bewegung von A nach B wird durch eine Kurve $C \subset \mathbb{R}^2$ definiert. (obwohl eine einfache Kurve).

$\forall p \in C$ ist p frei, d.h. Freiheit(p) $\geq r$

Betr. Abb. $\mathcal{D}: FP \rightarrow VD(S)$ bea: $C \subset FP$ ($\mathcal{D}(C) = C$)

die durch Schritte 2 bis 3 des Algorithmus definiert ist.

$\mathcal{D}(p) = p'$. \mathcal{D} = Streckung o. Projektion

Betr. Lots von p auf Segmente (mit min. Abstand)

Schnittpkt mit Vor. kante ist p'

Bea: Jetzt wollen wir noch nicht, dass C auf Vor.kanten liegt, wollen genau das zeigen

\Rightarrow a) \mathcal{D} ist stetig

$\Rightarrow \mathcal{D}(C)$ auch Kurve.

b) Freiheit($\mathcal{D}(p)$) \geq Freiheit(p) $\geq r$

$\min_{s \in S} |\mathcal{D}(p)-s| \geq \min_{s \in S} |p-s|$ weil wir immer den Bl. Abstand nehmen.

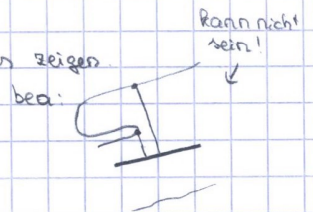
p haben zuerst gesucht, $p \in C \Rightarrow$ Beh.

$\geq r$ auch klar da Bew. \exists .

$\Rightarrow \forall p \in C: \mathcal{D}(p)$ ist frei und $\mathcal{D}(p) \in VD(S)$ weil $W(\mathcal{D}) = VD(S)$

$\Rightarrow \mathcal{D}$ ist eine legale Bewegung

Bea: $\mathcal{D}(A) = A' \wedge \mathcal{D}(B) = B'$ nach Alg.



4.2.11 Laufzeit:

Schritt 1: $O(n \log n)$ durch Plane Sweep.

Schritt 2: $O(n)$ lin. Suche in S

Schritt 3: $O(n)$ lin. Such auf $VD(S)$, P_A berechnen $\rightarrow O(1)$, A' berechnen $\rightarrow O(1)$, e_1, e_2 aufteilen und Freiheit(e_1), Freiheit(e_2) berechnen $\rightarrow O(1)$

Überprüfen ob A' bzw B' Vor. Knoten $\rightarrow O(1)$, dann lin. Suche bei den gespeicherten Voronoi-Knoten.

Schritt 4: $O(n)$ Pfadsuche in $VD(S)$ (z.B. DFS)

Schritt 5: $O(n)$ Ausgabe des Pfades.

4.2.12 Satz: Bewegungsplanungsproblem für eine Kreisscheibe in einer Szene von n Liniensegmenten in der Ebene kann in Zeit $O(n \log n)$ und Platz $O(n)$ gelöst werden.

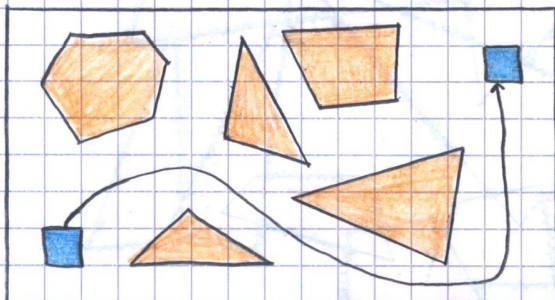
Beweis: s.o.

weil in einzelnen 5 Schritten immer Platz $O(n)$.

4.3 Problem 2: R ist konv. Polygon und S Menge von konv. Polygonen.

4.3.1. Problemschilderung: S ist eine Szene von m konvexen Polygonen P_1, \dots, P_m mit $n := \sum_{i=1}^m \# \text{Ecken von } P_i$ und $P_i \cap P_j = \emptyset \quad \forall i \neq j$

R (Roboter) ist konvexes Polygon mit c Ecken ($c = \text{Konstante}$),
Bewegungen: Nur Translationen von R (Keine Rotationen).



4.3.2. Ann:

- S ist in einem Rechteck eingeschlossen, z.B. Dummi-Hindernisse.
- Sei p ein beliebiger Referenzpunkt im Inneren von R , dann ist die Position von R durch die Position von p im \mathbb{R}^2 definiert.

4.3.3. Idee:

Reduziere das Problem auf das Bewegungsplanungsproblem eines plattformigen Roboters (p) in einer komplizierten Szene S' .

Dazu blähen wir alle Hindernisse P_1, \dots, P_m auf.

D.h. alle Hindernisse werden um das Maß vergrößert, auf das der Roboter verkleinert werden muss, um ein Pfad zu werden.

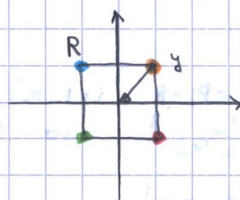
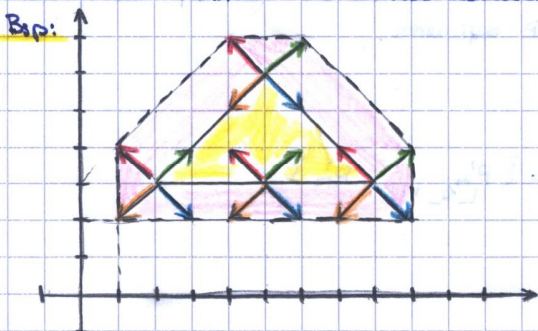
4.3.4. Konstruktion von P_i' (= aufgeblähtes Hindernis).

$P_i' = P_i - R =$ "Minkowski-Differenz".

$$= \{ x - y : x \in P_i \wedge y \in R \}$$

\uparrow \mathbb{R}^2 \uparrow \mathbb{R}^2 im Koordinatensystem des Roboters.

Bsp:



orange $(2,3) - (1,1) = (1,2)$

blau $(2,3) - (-1,1) = (3,2)$

rot $(2,3) - (1,-1) = (1,4)$

grün $(2,3) - (-1,-1) = (3,4)$

rosa kommt dazu

Betrachte die Szene $S' = \{P_1', \dots, P_m'\}$ mit $P_i' = P_i - R, i=1..m$.

Dann ist die Menge der freien Platzierungen von R in S' (bzgl. des Referenzpunktes p):
 $FP = \mathbb{R}^2 \setminus \bigcup_{i=1}^m P_i'$

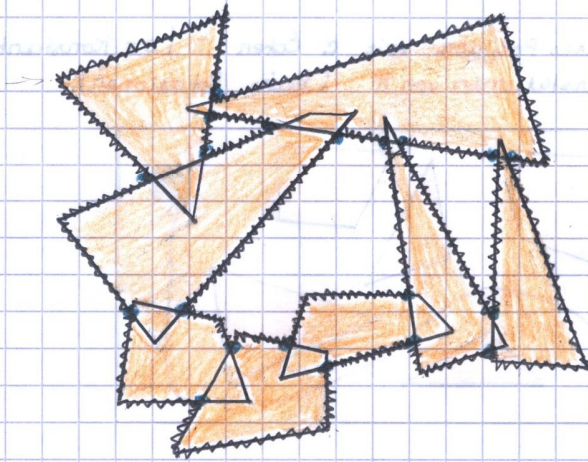
d.h. das Komplement der Vereinigung aller aufgeblähten Hindernisse.
FP ist i.a. nicht mehr zusammenhängend.

D.h. FP ist ein u.U. nicht zusammenhängendes Gebiet der Ebene, begrenzt durch Polygonzüge.

Es ist offensichtlich, dass eine Bewegung von A nach B genau dann möglich ist, wenn A und B in derselben Zusammenhangskomponente von FP liegen.

Übung 8.1.

4.3.5. Beispiel:



$n = \#$ alle Ecken
 Hier: $28 = n$
 Auf dem Rand dann: $19 < n$.

4.3.6. Wkt: der Rand oder die Kontur von FP hat Größe $O(n)$

Vermutung: $\#$ alle Ecken auf dem Rand der Vereinigung ist $O(n)$.

4.3.7. Satz: Seien P_1, \dots, P_m paarweise disjunkte, konvexe Polygone mit insgesamt n Ecken und R ein konvexes Polygon mit konstant vielen Ecken.

Dann hat $\text{Kontur} = K = \bigcup_{i=1}^m (P_i - R)$ $O(n)$ Ecken.
 Polygonfläche mit Höchern.

Bew. später.

4.3.8. Algorithmus:

Um Problem 2 zu lösen, müssen wir FP und seine Zusammenhangskomponenten berechnen. Dazu berechnen wir zunächst die Kontur (K) von FP, dh. die Menge der Kanten, die den Rand von FP definieren.

→ Divide & Conquer.

Algorithmus:

1) Berechne $S' = \{P_i' = P_i - R : i = 1..m\}$.

2) Sei $S_1 := \{P_1' \dots P_{\lfloor m/2 \rfloor}'\}$ und $S_2 := \{P_{\lfloor m/2 \rfloor + 1}' \dots P_m'\}$

Falls $|S_1| = 1 \Rightarrow K_1 = P_1'$

Falls $|S_2| = 1 \Rightarrow K_2 = P_{\lfloor m/2 \rfloor + 1}'$

sonst:

Berechne rekursiv:

$K_1 = \text{Kontur von } S_1$

$K_2 = \text{Kontur von } S_2$

3) Berechne $K = \text{Kontur von } S'$ durch Überlagerung von K_1 und K_2 .

4.3.9. Laufzeit: Schritt 1: $O(n)$

Dazu nötig, dass Minkowski-Differenz von konvexen Polygonen in linearer Zeit.

→ Übung

Rest: $T(n) = 2 \cdot T(n/2) + \text{Zeit für Mischen}$

↑ Vereinigung von zwei Konturen K_1 und K_2 .

