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.

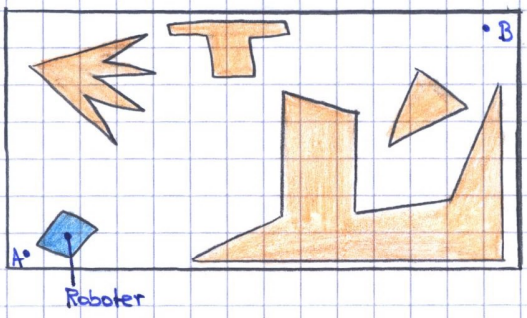
Aufgabe: Beantworte die Frage, ob R von A nach B bewegt werden kann, ohne mit Hindernissen aus S zu kollidieren bzw. gib einen kollisionsfreien Weg an.

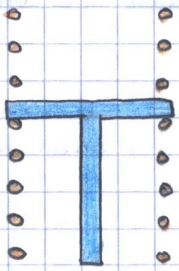
→ Piano Moves Problem.

4.1.2. Bemerkungen:

1) Wir behandeln zwei Spezialfälle:
   - S ist Menge von Kurvensegmenten u. R ist eine Kreisscheibe.
   - S ist Menge von konvexen Polygone, R ist konvexes Polygon.
2) Bewegung: nur Translationen.

Es gibt Vielzahl weiterer Varianten:
- Rotationen erlaubt
- Kürzester Weg.
- Sichester Weg.





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

4.2.1. Idee: Versuche den sichersten Weg zu finden, dh. 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. VD(S)

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

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

Anschließend Entfernung 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.

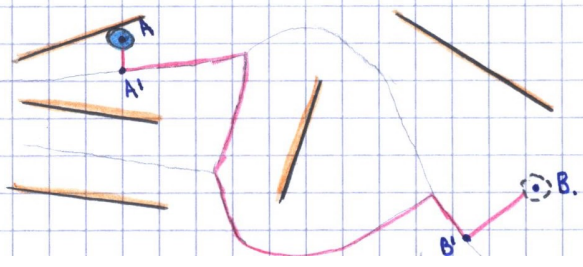
b) p heißt frei $\Leftrightarrow$ Freiheit (p) $\geq r$, wobei r = Radius von R.
(dh. Kreisscheibe R kann mit Mittelpkt auf p plaziert werden, ohne ein Segment zu schneiden).

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

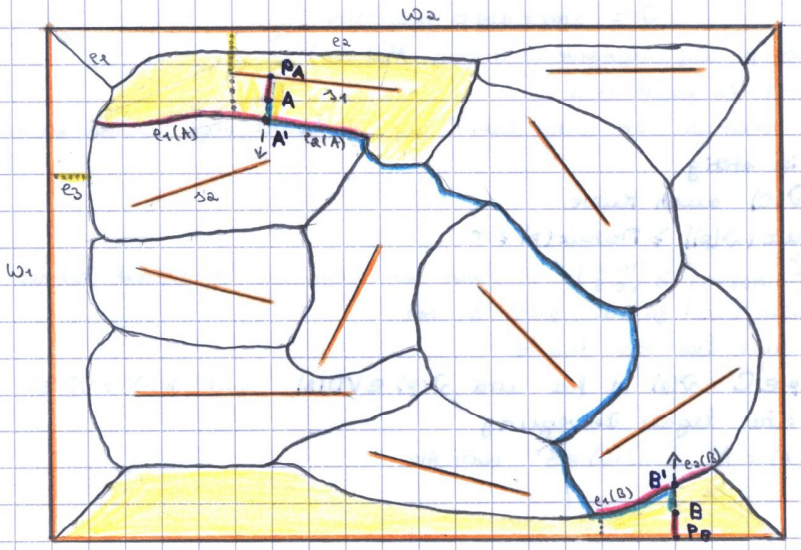
4.2.7 Beispiel:



4.2.8 Algorithmus:

1. Konstruiere VD(S). Speichere für jede Kante die beiden Orte, die e definieren. (entweder Segmentendpkte oder Segmentwand).
   Und Freiheit(e) := minimaler Abstand eines Pktes auf e zu einem Segment.
   (Freiheit(e) ≥ r ⇒ 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 (→ Point Location)
   (P_A, P_B sind Schnittpkte des Lots von A bzw. B auf nächstes Segment).
   Dh: Betrachte Lots von A bzw. B auf alle Segmente in S, suche nach einem Lot, der eine Voronoi-Kante schneidet → lin. Suche.
3. A' ← erster Schnittpkt des Strahls $\overrightarrow{P_A A}$ mit VD(S)
   B' ← 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.
   → Berechne Freiheit(e_1), Freiheit(e_2)!
4. Suche einen Pfad P in VD(S) von A' nach B' mit Freiheit(e) ≥ r ∀ e ∈ P.
   z.B. durch DFS oder BFS.
   Falls kein solcher Pfad existiert, melde "keine Lösung".
   → STOP.
5. Der Weg A → P → B ist eine mögliche Bewegung.

4.2.9 Bsp:



1. für e_1 speichere: (ω_1, ω_2) — Freiheit(e_1) = 0 cm
   für e_2 speichere: (s_1, ω_2) — Freiheit(e_2) = 0,5 cm
   für e_3 speichere: (Endpkt von s_2, ω_1) — Freiheit(e_3) = 0,5 cm. usw. ...
2. Berechne Lots von A und B auf alle Segmente (ind. 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 gelb.
3. Teile e(A) in $e_1(A)$ und $e_2(A)$ und e(B) in $e_1(B)$ und $e_2(B)$
   Freiheit($e_1(A)$) = 0,6 cm, Freiheit($e_2(A)$) = 0,7 cm
   Freiheit($e_1(B)$) = 0,3 cm, Freiheit($e_2(B)$) = 1 cm.
4. ∀ e ∈ blau gilt Freiheit(e) ≥ 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
$\Rightarrow p_A \in S_v \in S$
Seien A, B frei $\Rightarrow$ Freiheit(A) $\geq r \wedge$ Freiheit(B) $\geq r$
$\text{Freiheit}(A) := \min_{s \in S} |A - s|$, $\text{Freiheit}(B) := \min_{s \in S} |B - s|$

Es gibt: $r \leq \text{Freiheit}(A) \leq \text{Freiheit}(p) \quad \forall p \in \overline{AA'}$
$\Rightarrow$ Es gibt eine legale Bewegung von R von A nach A'
weil $\text{Freiheit}(p) \geq r \quad \forall p \in \overline{AA'}$.
$B \to B'$ 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. (oBdA eine einfache Kurve).
$\forall p \in C$ ist p frei, dh. $\text{Freiheit}(p) \geq r$
Betr. Abb. $\vec{v}: \mathbb{R}P \to VD(S)$ bea: $C \subset \mathbb{R}P$ $(D(\vec{v}) = C)$
die durch Schritte 2 bis 3 des Algorithmus definiert ist.
$\vec{v}(p) = p'$. $\vec{v}$ = Streckung o projektion
Betr. Lots von p auf Segmente (mit min. Abstand)
Schnittpkt mit Vor.kante ist p'.
Bea: Jetzt wissen wir noch nicht, dass C auf Vor.kanten liegt, wollen genau das zeigen

$\Rightarrow$ a). $\vec{v}$ ist stetig
$\Rightarrow \vec{v}(C)$ auch Kurve.
b) $\text{Freiheit}(\vec{v}(p)) \geq \text{Freiheit}(p) \geq r$
$\min''_{s \in S} |\vec{v}(p) - s| \geq \min_{s \in S} |p - s|$ weil wir immer den kl. Abstand nehmen.
p haben zuerst gesucht, $p \in C \Rightarrow$ Beh.
$\geq r$ auch klar da Bew. $\exists$.
$\Rightarrow \forall p \in C$: $\vec{v}(p)$ ist frei und $\vec{v}(p) \in VD(S)$ weil $W(\vec{v}) = VD(S)$
$\Rightarrow \vec{v}$ ist eine legale Bewegung
Bea: $\vec{v}(A) = A' \wedge \vec{v}(B) = B'$ 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 $VD(S)$, $\overrightarrow{p_A A}$ berechnen $\to O(1)$, A' berechnen $\to O(1)$, e_1, e_2 aufteilen und $\text{Freiheit}(e_1)$, $\text{Freiheit}(e_2)$ Berechnen $\to O(1)$
Überprüfen ob A' bzw B' Vor. Knoten $\to O(n)$, denn lin. Suche bei den gespeicherten Voronoiknoten.
Schritt 4: $O(n)$ Pfadsuche in $VD(S)$ (ZB DFS)
Schritt 5: $O(n)$ Ausgabe des Pfades.

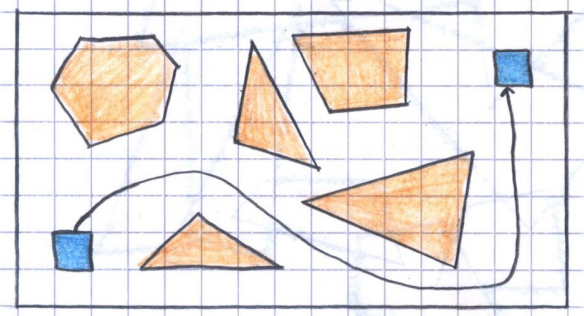
4.2.12. Satz: Bewegungsplanungsproblem für eine Kreisscheibe in einer Szene von n Liniensegmenten in der Ebene kann in Zeit $O(n \log n)$ und Platz $O(n)$ gelöst werden.
(Platz $O(n)$: weil in einzelnen 5 Schritten immer Platz $O(n)$.)

Beweis: s.o.

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 $P_1 .. P_m$ mit $n := \sum_{i=1}^{m}$ # 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 = Konstante).
Bewegungen: Nur Translationen von R (keine Rotationen).



4.3.2 Anm:

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

4.3.3. Idee:

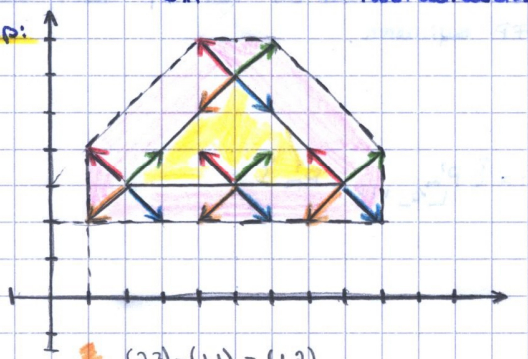
Reduziere das Problem auf das Bewegungsplanungsproblem eines pktförmigen Roboters (p) in einer komplizierten Szene S'.
Dazu blähen wir alle Hindernisse $P_1 .. P_m$ auf.
D.h. alle Hindernisse werden um das Maß vergrößert, auf das der Roboter verkleinert werden muß, um ein Pkt zu werden.

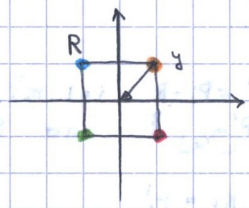
4.3.4. Konstruktion von P_i' (= aufgeblähtes Hindernis).

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

$= \{ x-y : \underbrace{x \in P_i}_{\in \mathbb{R}^2} \wedge \underbrace{y \in R}_{\text{im Koordinatensystem des Roboters.}} \}$

Bsp:





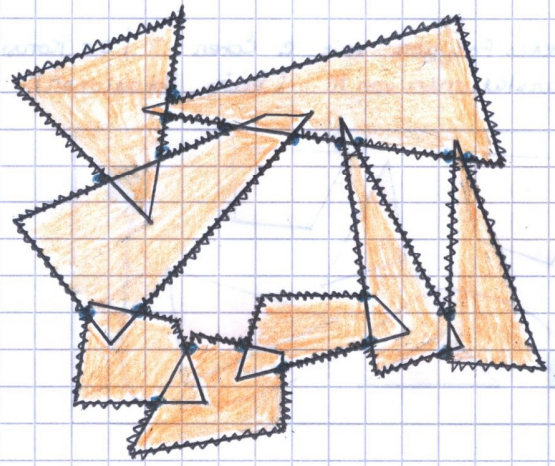
Übung ! 8.1.

- $(2,3) - (1,1) = (1,2)$
- $(2,3) - (-1,1) = (3,2)$
- $(2,3) - (1,-1) = (1,4)$
- $(2,3) - (-1,-1) = (3,4)$
- Kommt dazu

Betrachte die Szene $S' = \{P_1' \ldots P_m'\}$ mit $P_i' = P_i - R$, $i = 1..m$.
Dann ist die Menge der freien Plazierungen von R in S' (bzgl. des Referenzpktes 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 zushgd.
D.h. FP ist ein u.U. nicht zushgdes 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.

4.3.5. Beispiel:



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

4.3.6 Lemma: 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 $P_1..P_m$ paarweise disjunkte, konvexe Polygone mit insgesamt n Ecken und R ein konvexes Polygon mit konstant vielen Ecken.
Dann hat Kontur = $K = \underbrace{\bigcup_{i=1}^{m} (P_i - R)}_{\text{Polygonfläche mit Löchern}}$ O(n) Ecken.

Beweis 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 $S_2 := \{P'_{\lfloor m/2 \rfloor + 1}, .. P_m'\}$

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 = Kontur von S_1
K_2 = Kontur von S_2
3) Berechne K = 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: $T(n) = 2 \cdot T(n/2) +$ Zeit für Mischen
↑ Vereinigung von zwei Konturen K_1 und K_2.