# Übungsblatt 4

Julius Auer, Alexa Schlegel

**Aufgabe 1** (Bewegungsplanung in der Ebene):

Gegeben sei ein kreisförmiger Roboter $R$ mit Radius $d$, sowie eine Menge $H = p_1, \ldots, p_n$ von $n$ punktförmigen Hindernissen in der Ebene. Eingabewerte sind Startpunkt $s$ und Zielpunkt $z$. Die Ausgabe ist kollisionsfreier Weg $w$ (Polygonzug), falls er existiert, sonst $\emptyset$.

Die Idee des Algorithmus ist es, aus $H$ ein Voronio-Diagramm (VD) zu erstellen und das Zentrum des Roboters auf den Voronio-Kanten (VK) entlang zum bewegen. Zusätzlich werden diejenigen VK entfernt, wo der Abstand der zugehörigen Punkte kleiner ist als der Durchmesser des Roboter. Auf den verbleibenden Kanten kann sich der Roboter bewegen, d.h. sein Zentrum bewegt sich.

Der Algorithmus wird in 3 Schritte unterteilt:

① Roboter von $s$ zum/auf VD bewegen
② Roboter entlang des VD bewegen
② Roboter von VD zu $z$ bewegen

**Algorithm 1** movingAround($H, s, z, d$)

1: Berechnung VD($H$)
2: VR($s$) und VR($z$) bestimmen
3: **if** $s$ bzw. $z$ kollidieren mit Punkt in VR($s$) bzw. VR($z$) **then**
4: **return** $\emptyset$
5: **end if**
6: $s'$ und $z'$ auf VD finden und als Knoten zu VD($H$) hinzufügen und Kanten aufsplitten (senkrecht vom Punkt in zugehöriger VR wegbewegen bis VK getroffen wird, siehe Abbildung1)
7: Sei $G$ der zugehörige Graph zum VD($H$): alle Kanten entfernen, wo der Abstand der zugehörigen Punkte $\leq d$ ist
8: **if** Weg $w$ von $s'$ nach $z'$ existiert **then**
9: **return** $w \cup \overline{ss'}$ und $\overline{pp'}$
10: **else**
11: **return** $\emptyset$
12: **end if**

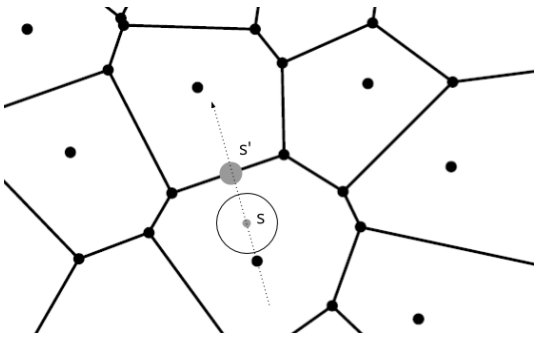


Abbildung 1: Finden von $s'$

Der existierende Weg kann z.B. per Tiefensuche oder Dijkstra bestimmt werden. Liegt der Start bzw. Endpunkt auf einer Kante die später entfernt wird, so liefert die Suche nach einem existierenden Weg kein Ergebnis.

Was mir noch nicht ganz klar ist, wenn $s$ und $z$ in einer unbeschränkten VR liegen (bzw. man da immer irgendwie hin kommt) dann könnte man doch immer einmal komplett außen rum laufen, oder? Oder ist man in einem Zimmer gefangen wo die Punkte auf dem Boden rumliegen und man sonst gegen die Wand läuft?

**Aufgabe 2** (Geometrische Graphen):

a) Induktion über $|V|$:

I.A.: Für $G_1 = (V, E, F)$ zusammenhängend und planar mit $|V| = 1$ folgt offensichtlich $|E| = 0$ und $|F| = 1$.

I.V.: Für $G_n = (V, E, F)$ zusammenhängend und planar mit $|V| = n$ gilt: $|E| = |V| + |F| - 2$

$n \to n + 1$: Aus $G_{n+1} = (V', E', F')$ zusammenhängend und planar mit $|V'| = n + 1$ werden ein Knoten $v$ und alle zu $v$ inzidenten Kanten entfernt. Der entstehende Graph muss nicht zusammenhängend sein. Unabhängig von der Anzahl der Zusammenhangskomponenten (ZHKs) lässt sich aus den folgenden Beobachtungen $|F'|$ ableiten:

① $G_{n+1}$ ohne $v$ und die zu $v$ inzidenten Kanten besteht aus $k \leq deg(v)$ planaren ZHKs $G^1, ..., G^k$. Für jede ZHK gilt die I.V..
② Für das Verbinden dieser ZHKs waren zuvor $k$ Kanten erforderlich, die keine Facetten begrenzten (sonst hätte es einen Kreis und nicht mehrere ZHKs gegeben). Die anderen $deg(v)-k$ Kanten bildeten $deg(v) - k$ Facetten (durch das Hinzufügen zu einer bestehenden ZHK muss ein Kreis und somit eine neue Facette entstehen).
③ Ferner ist klar, dass sich die $k$ ZHKs eine Facette (die "Äußere") teilen - also bei ① $k - 1$ Facetten zuviel gezählt werden.
Somit ergibt sich insgesamt für $|F'|$:

$$\begin{aligned}
|F'| &= \overbrace{\sum_{i=1}^{k} |F^i|}^{①} + \overbrace{(deg(v) - k)}^{②} - \overbrace{(k - 1)}^{③} \\
&= \sum_{i=1}^{k} (|E^i| - |V^i| + 2) + deg(v) - 2 \cdot k + 1 \\
&= |E'| - deg(v) - (|V'| - 1) + 2 \cdot k + deg(v) - 2 \cdot k + 1 \\
&= |E'| - |V'| + 2
\end{aligned}$$

□

b) Sei $D$ die Menge der Dreiecke des triangulierten Graphen $G = (V, E)$. $r$ Knoten liegen auf dem Rand der konvexen Hülle der Punkte aus $G$ und beschreiben einen geschlossenen Polygonzug, der folglich aus ebenfalls $r$ Kanten $E_r$ besteht. Jede dieser Kanten grenzt an genau ein Dreieck. Die $|E| - |E_r|$ Kanten, die im Inneren des Graphen liegen, begrenzen jeweils zwei Dreiecke. Jedes Dreieck wiederum wird von drei Kanten begrenzt. Somit gilt:

$$\begin{aligned}
|E_r| + 2 \cdot (|E| - |E_r|) &= 3 \cdot |D| \\
\Leftrightarrow |E| &= \frac{3 \cdot |D| + r}{2}
\end{aligned}$$

In (a) wurde gezeigt, dass:

$$|F| = |E| - |V| + 2$$

Wobei hier offensichtlich $|D| = |F| - 1$ (die "äußere" Facette ist kein Dreieck). Das sollte ausreichen, etwas Aussagekräftiges auszurechnen :)

$$\begin{aligned} |F| &= |E| - |V| + 2 \\ \Leftrightarrow |D| + 1 &= \frac{3 \cdot |D| + r}{2} - n + 2 \\ \Leftrightarrow |D| &= 2 \cdot n - r - 2 \\ &= 2 \cdot (n-1) - r \end{aligned}$$

□

**Aufgabe 3** (Voronoi-Diagramm):

Das dürfte straight-forward funktionieren:

Eingabe ist ein Voronoi-Graph mit Knoten $V$, Kanten $E$ und Facetten $F$ sowie die leere Menge $CH$ die nach aufruf von $CH(V, E, F, CH)$ die konvexe Hülle von $S$ enthält. Für eine Kante $e$ seien $e.f1$ und $e.f2$ die angenzenden Facetten. Für eine Facette $f$ seien $E(f)$ die angrenzenden Kanten.

**Algorithm 2** $CH(V, E, F, CH)$

```
1:  start := Facette mit minimaler x-Koordinate
2:  f := start
3:  last := null
4:  repeat
5:    CH ← f
6:    for each e ∈ E(f) do
7:      if e is unbound and not (e.f1 = last or e.f2 = last) then
8:        last := f
9:        f := (f = e.f1?e.f1 : e.f2)
10:       break
11:     end if
12:   end for
13: until r = start
```

Es wird auf den äußeren Facetten entlanggelaufen, bis man wieder den Ausgangspunkt erreicht. Hierfür überquert man immer genau die Kante von der man nicht gekommen ist, und die ein offenes Ende hat.

(1) benötigt $O(n)$ Zeit.
(4) wird $O(n)$ mal wiederholt
(6-12) kann zwar $O(n)$ mal wiederholt werden, insgesamt wird jedoch offensichtlich in (4-13) jede äußere (ungebundene) Kante nur konstant oft betrachtet. Die Anzahl Kanten ist asymptotisch begrenzt durch die Anzahl der Knoten, womit insgesamt keine Kosten von mehr als $O(n)$ zu erwarten sind.

Insgesamt ist ein Zeitaufwand von $O(n)$ zu erwarten, zusätzlicher Speicherplatz ist nicht erforderlich ($O(n)$ für die konvexe Hülle).