Vorlesung

Geometrische Algorithmen

Sichtbarkeitsgraphen und kürzeste Wege

Sven Schuierer

Überblick

1. Kürzeste Wege
2. Sichtbarkeitsgraphen
3. Berechnung des Sichtbarkeitsgraphen
4. Kürzeste Wege für polygonale Roboter

1 Kürzeste Wege

Vorlesung Bewegungsplanung:

Berechnung eines kollisionsfreien Weges

Andere Anforderungen:

- Geringe Länge
- Wenige Knicke
- Mindestkurvenradius

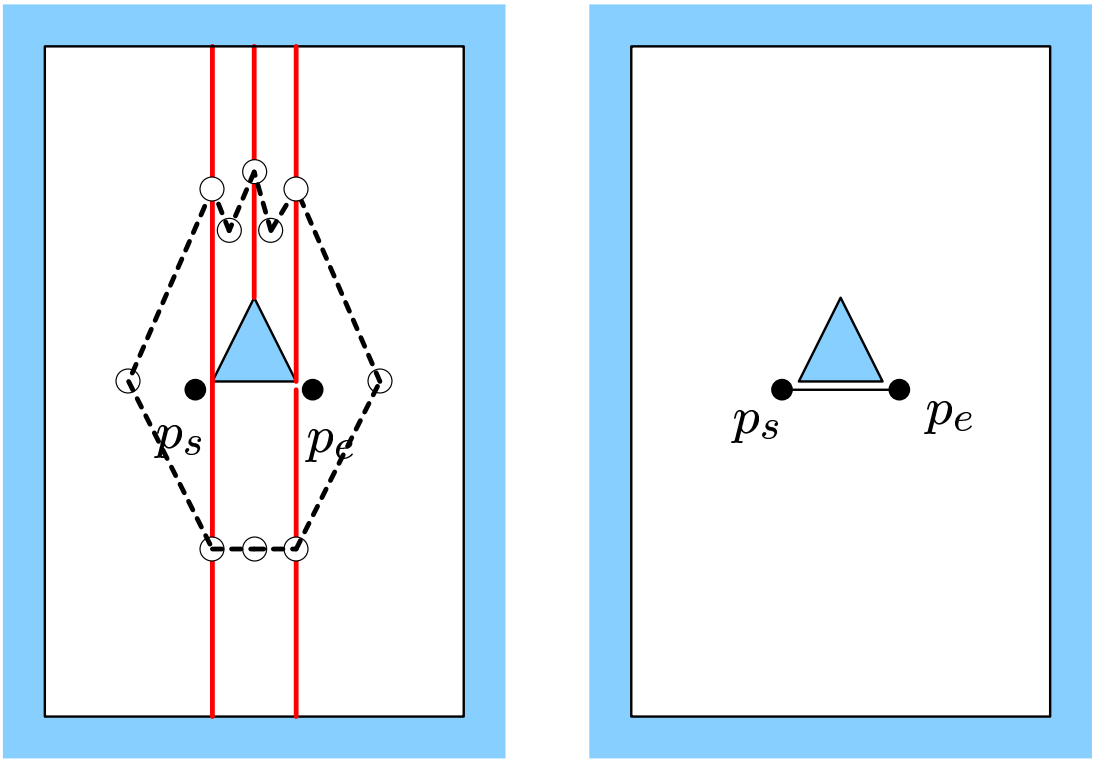
Kürzeste Wege für Punktroboter

Gegeben:

- Menge $\mathcal{H}$ von disjunkten polygonalen (offenen) Hindernissen mit insgesamt n Kanten
- Startposition p_s, Zielposition p_e, beide in $\mathcal{C}_{free}$

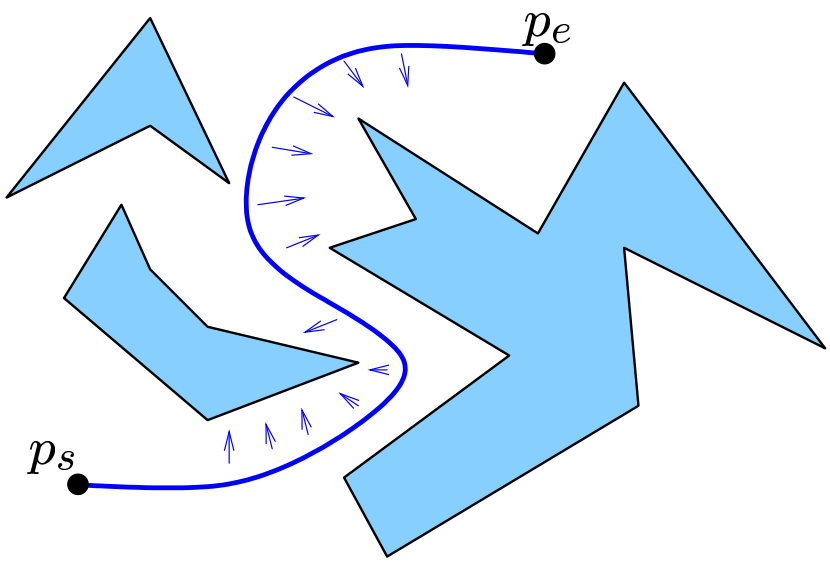
Gesucht:

Der kürzeste Weg von p_s nach p_e in $\mathcal{C}_{free}$



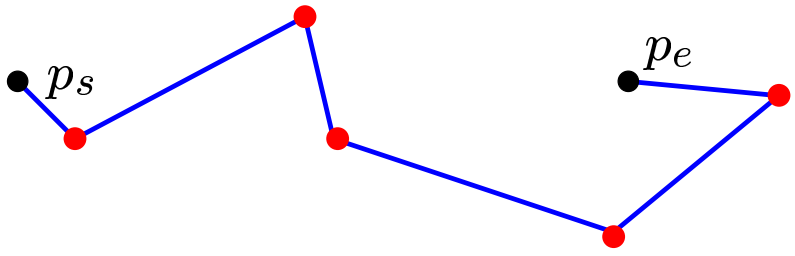
Eigenschaften kürzester Wege

Gummiband-Eigenschaft:



Definition

Sei W ein polygonaler Weg von p_s nach p_e. Dann heißen die Ecken von W, die ungleich p_s oder p_e sind, innere Ecken von W.



Eigenschaften kürzester Wege

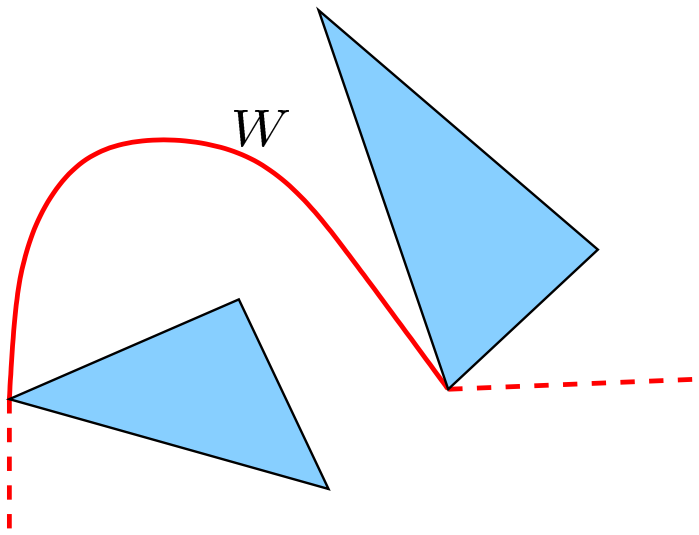
Lemma

Jeder kürzeste Weg W von p_s nach p_e zwischen einer Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen ist ein polygonaler Weg, dessen innere Ecken Ecken von $\mathcal{H}$ sind.

Beweis: :

Polygonalität:

Angenommen, W ist nicht polygonal
$\Rightarrow$ Es gibt ein $p \in W$ im Inneren von $\mathcal{C}_{free}$, so daß kein Liniensegment in W p enthält.
$\Rightarrow$ Es gibt einen Kreis in $\mathcal{C}_{free}$, der p enthält



Eigenschaften kürzester Wege

Beweis: : (Fortsetzung)

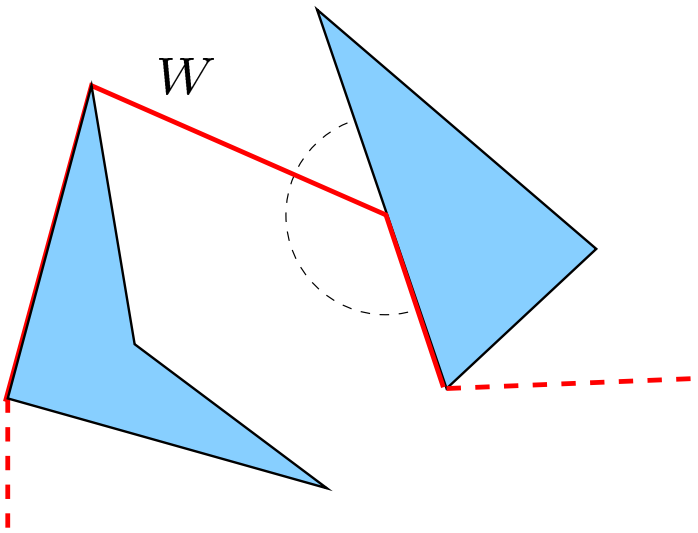
Ecke v von W:

Angenommen, v ist keine Ecke von $\mathcal{H}$

$\Rightarrow$ v ist nicht im Inneren von $\mathcal{C}_{free}$

$\Rightarrow$ v ist im inneren einer Kante

$\Rightarrow$ Es gibt einen Halbkreis in $\mathcal{C}_{free}$, der p enthält



Eigenschaften kürzester Wege

Korollar

Ein kürzester Weg von p_s nach p_e zwischen einer Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen besteht aus:

1. Liniensegmenten von p_s bzw. p_e zu Ecken von $\mathcal{H}$ und
2. Liniensegmenten von einer Ecke von $\mathcal{H}$ zu einer anderen.

Definition

Zwei Punkte v_1 und v_2 heißen gegenseitig sichtbar, falls das Liniensegment $\overline{v_1 v_2}$ nicht das Innere eines Hindernisses schneidet.

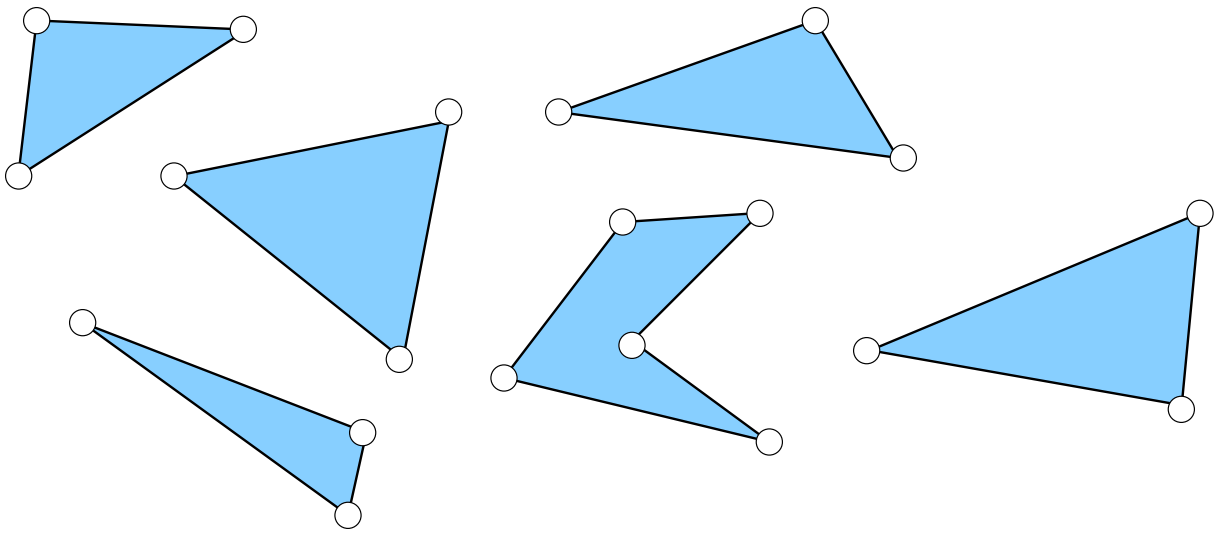
2 Sichtbarkeitsgraphen

Sichtbarkeitsgraph $\mathcal{G}_{vis}(\mathcal{H})$ von $\mathcal{H}$:

Knoten $\mathcal{V}_{vis}(\mathcal{H})$ von $\mathcal{G}_{vis}(\mathcal{H})$: Ecken von $\mathcal{H}$

Kanten $\mathcal{E}_{vis}(\mathcal{H})$ von $\mathcal{G}_{vis}(\mathcal{H})$:

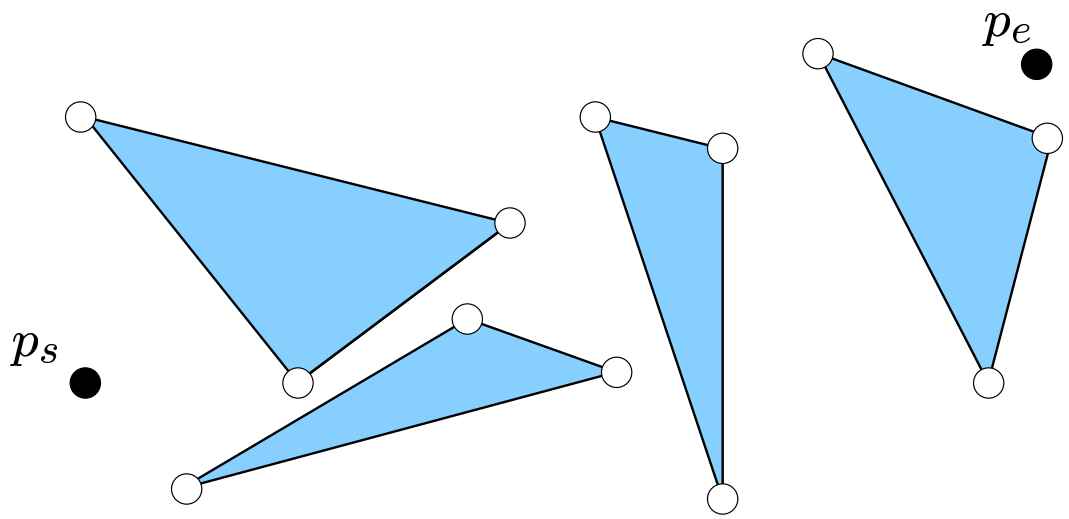
$(v_1, v_2) \in \mathcal{E}_{vis}(\mathcal{H})$, falls v_1 und v_2 gegenseitig sichtbar sind.



Sichtbarkeitsgraphen und kürzeste Wege

Korollar

Ein kürzester Weg von p_s nach p_e zwischen einer Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen besteht aus Kanten des Sichtbarkeitsgraphen $\mathcal{G}_{vis}(\mathcal{H}^*)$, wobei $\mathcal{H}^* = \mathcal{H} \cup \{p_s, p_e\}$.



Berechnung kürzester Wege

Algorithmus *kürzesterWeg*

Input: Eine Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen und zwei Punkte p_s und p_e in $\mathcal{C}_{free}$

Output: Der kürzeste kollisionsfreie Weg von p_s nach p_e

1 $\mathcal{G}_{vis} \leftarrow$ Sichtbarkeitsgraph$(\mathcal{H} \cup \{p_s, p_e\})$
2 Gewichte jede Kante (u, v) in $\mathcal{G}_{vis}$ mit der euklidischen Distanz von u nach v
3 Berechne einen kürzesten Weg W von p_s nach p_e in $\mathcal{G}_{vis}$ mit dem Algorithmus von Dijkstra
4 Gebe W zurück

Analyse

3 Berechnung des Sichtbarkeitsgraphen

Naives Verfahren:

Algorithmus *Sichtbarkeitsgraph1*

Input: Eine Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen

Output: Der Sichtbarkeitsgraph $\mathcal{G}_{vis}(\mathcal{H})$ von $\mathcal{H}$

1 $\mathcal{G}_{vis} \leftarrow (\mathcal{V}_{vis}, \mathcal{E}_{vis})$ mit
2 $\quad \mathcal{V}_{vis} \leftarrow$ die Menge der Eckpunkte von $\mathcal{H}$
3 $\quad \mathcal{E}_{vis} \leftarrow \emptyset$
4 **for all** $(u, v) \in \mathcal{V}_{vis} \times \mathcal{V}_{vis}$ **do**
5 $\quad$ **if** $\overline{uv}$ schneidet kein Hindernis
6 $\quad\quad$ **then** füge (u, v) zu $\mathcal{E}_{vis}$ hinzu
7 Gebe $\mathcal{G}_{vis} = (\mathcal{V}_{vis}, \mathcal{E}_{vis})$ zurück

Berechnung des Sichtbarkeitsgraphen

Besserer Algorithmus:

Algorithmus *Sichtbarkeitsgraph2*

Input: Eine Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen

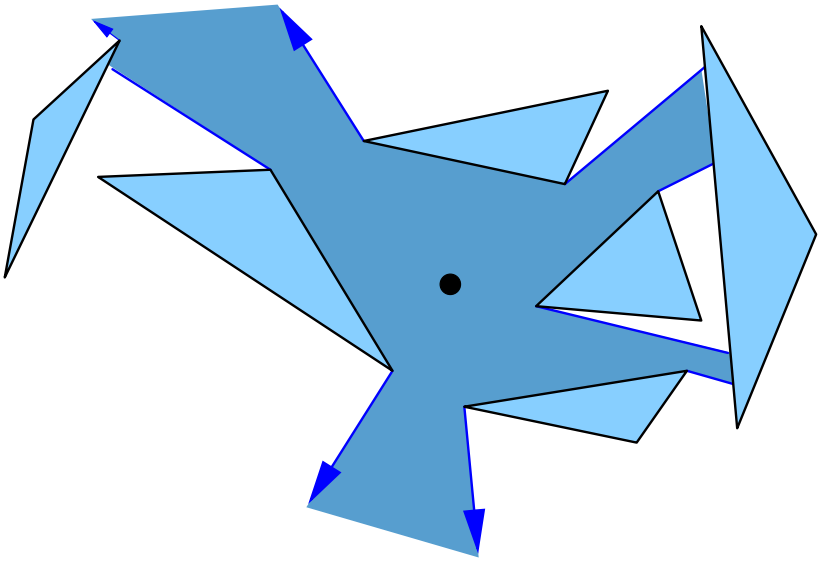
Output: Der Sichtbarkeitsgraph $\mathcal{G}_{vis}(\mathcal{H})$ von $\mathcal{H}$

1 $\mathcal{G}_{vis} \leftarrow (\mathcal{V}_{vis}, \mathcal{E}_{vis})$ mit
2 $\quad \mathcal{V}_{vis} \leftarrow$ die Menge der Eckpunkte von $\mathcal{H}$
3 $\quad \mathcal{E}_{vis} \leftarrow \emptyset$
4 **for all** $v \in \mathcal{V}_{vis}$ **do**
5 $\quad W \leftarrow$ SichtbareEcken$(v, \mathcal{H})$
6 $\quad$ **for all** $w \in W$ **do**
7 $\quad\quad$ füge die Kante (v, w) zu $\mathcal{E}_{vis}$ hinzu
8 Gebe $\mathcal{G}_{vis} = (\mathcal{V}_{vis}, \mathcal{E}_{vis})$ zurück

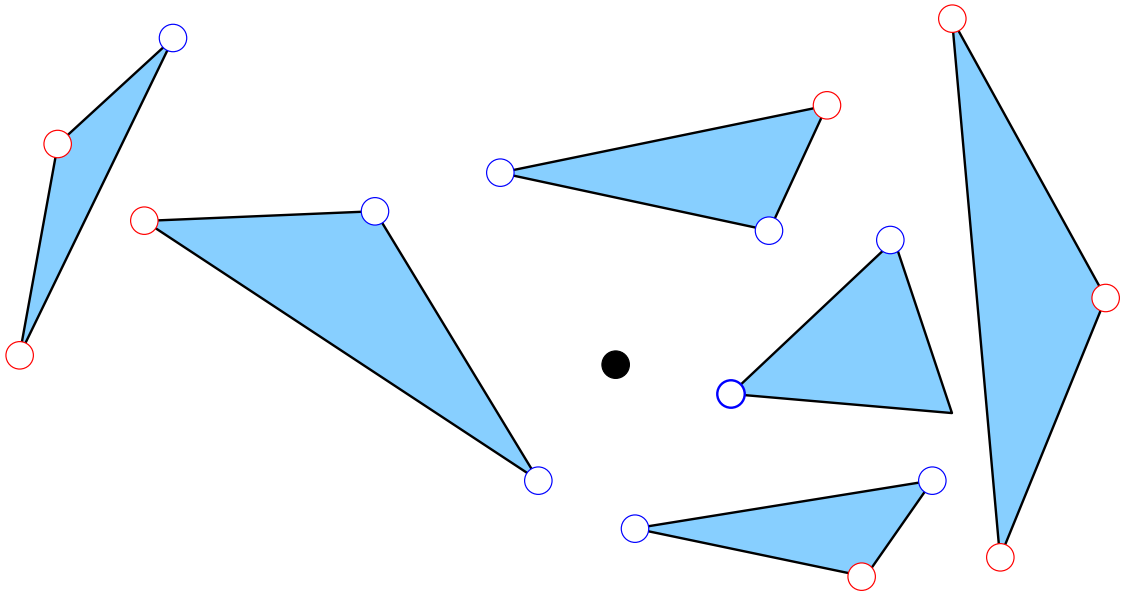
Sichtbarkeitspolygon

Definition

Die Menge der Punkte, die von einem Punkte p der Ebene aus sichtbar sind, heißt das Sichtbarkeitspolygon von p.



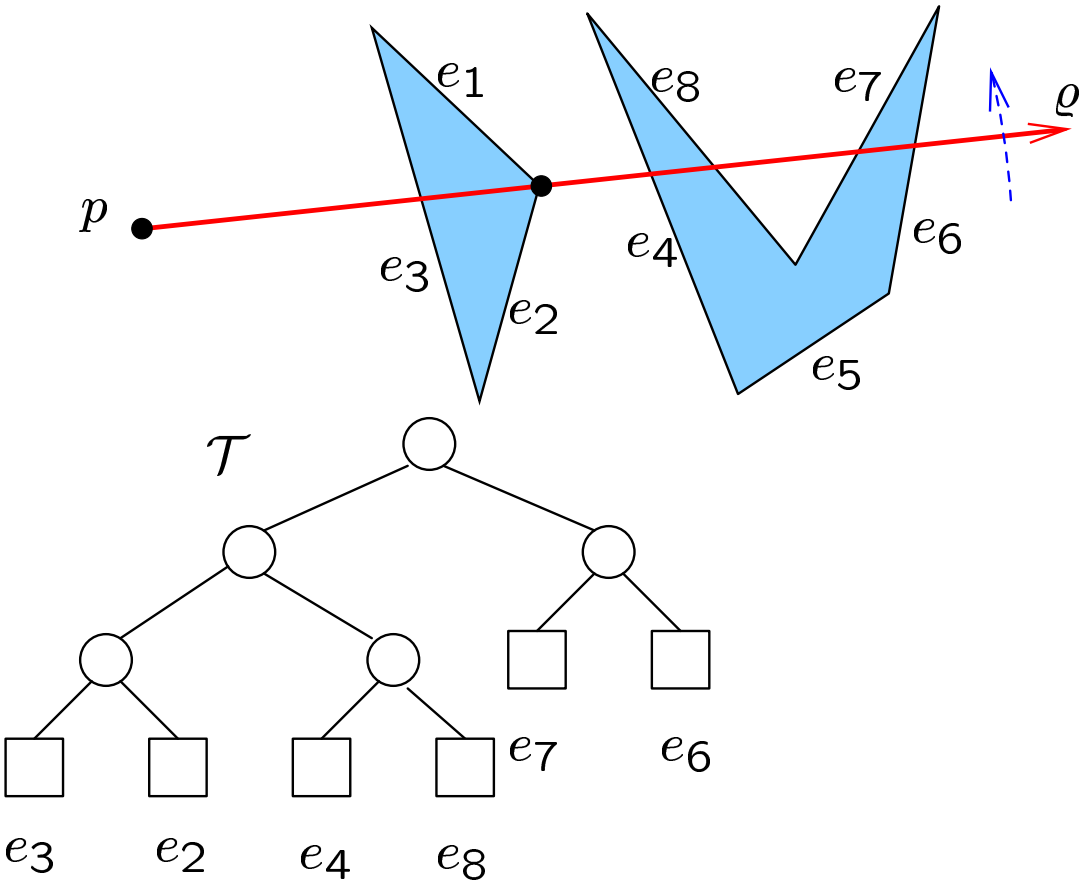
Hier: von p aus sichtbare Ecken



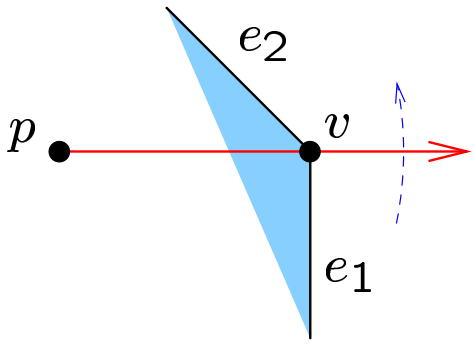
Berechnung der sichtbaren Ecken

Idee:

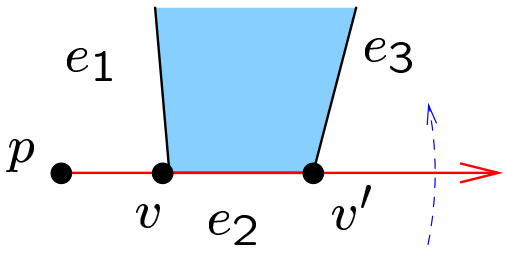
- Sortiere die Ecken zyklisch um p
- Betrachte die Ecken in dieser Reihenfolge mit Hilfe eines Scanstrahls ϱ
- Verwalte die Kanten, die von ϱ geschnitten werden, in einer Suchstruktur



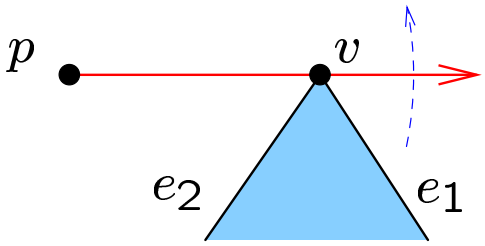
Rotationsweep Events



$sichtbar(v)$
$delete(e_1)$
$insert(e_2)$



$sichtbar(v)$
$sichtbar(v')$
$insert(e_1)$
$insert(e_3)$



$sichtbar(v)$
$delete(e_1)$
$delete(e_2)$

Sichtbare Ecken

Algorithmus *SichtbareEcken*

Input: Menge $\mathcal{H}$ von Polygonen und Punkt p nicht im Innern eines Polygons

Output: Die von p aus sichtbaren Ecken in $\mathcal{H}$

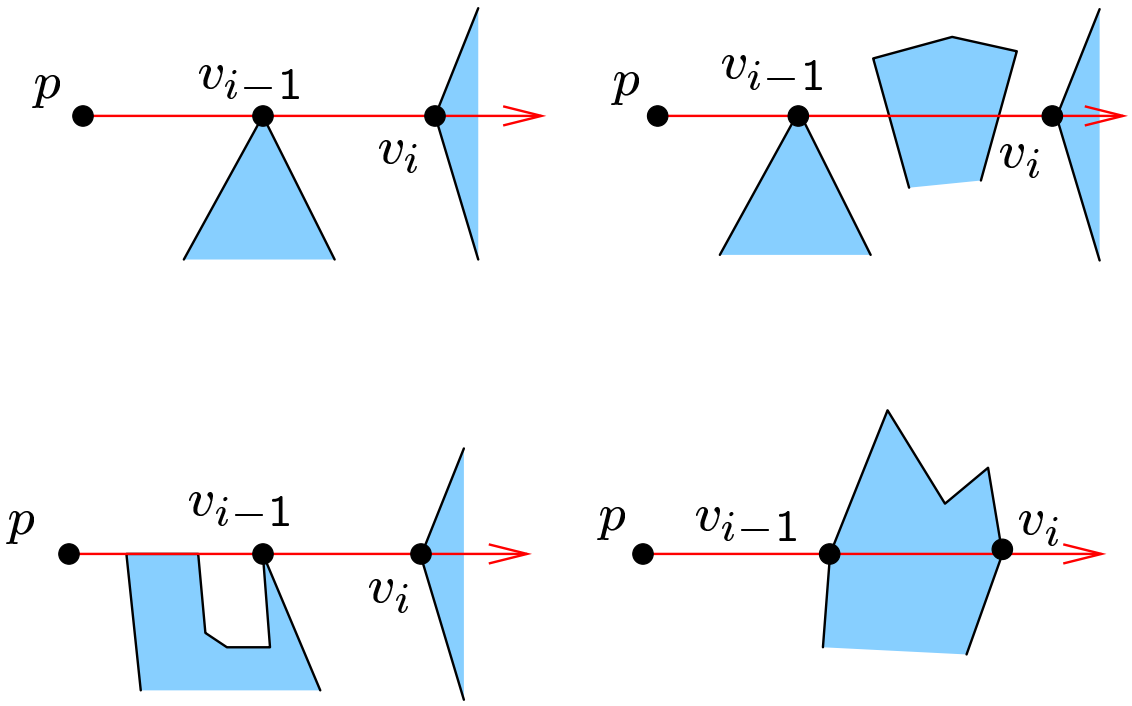
1 Sortiere die Ecken v nach $winkel(p, v)$ (im Fall gleicher Winkel nach Abstand zu p)
$(v_1, \ldots, v_n) \leftarrow$ sortierte Reihenfolge
2 Sei ϱ der horizontale Strahl beginnend bei p
3 Finde alle Kanten e die ϱ schneiden und speichere sie in einem Suchbaum $\mathcal{T}$
4 $W \leftarrow \emptyset$
5 **for** $i \leftarrow 1$ **to** n **do**
6 **if** $sichtbar(v_i)$
7 **then** $W \leftarrow W \cup \{v_i\}$
8 füge die zu v_i inzidenten Kanten links von ϱ in $\mathcal{T}$ ein
9 entferne die zu v_i inzidenten Kanten rechts von ϱ aus $\mathcal{T}$
10 **return** W

Sichtbarkeit von Ecken

Sichtbarkeit von v_i ist abhängig von:

- Suchstruktur $\mathcal{T}$
- Ecke v_{i-1}

Verschiedene Fälle: v_{i-1} ist sichtbar



Sichtbarkeit von Ecken

Algorithmus *sichtbar*

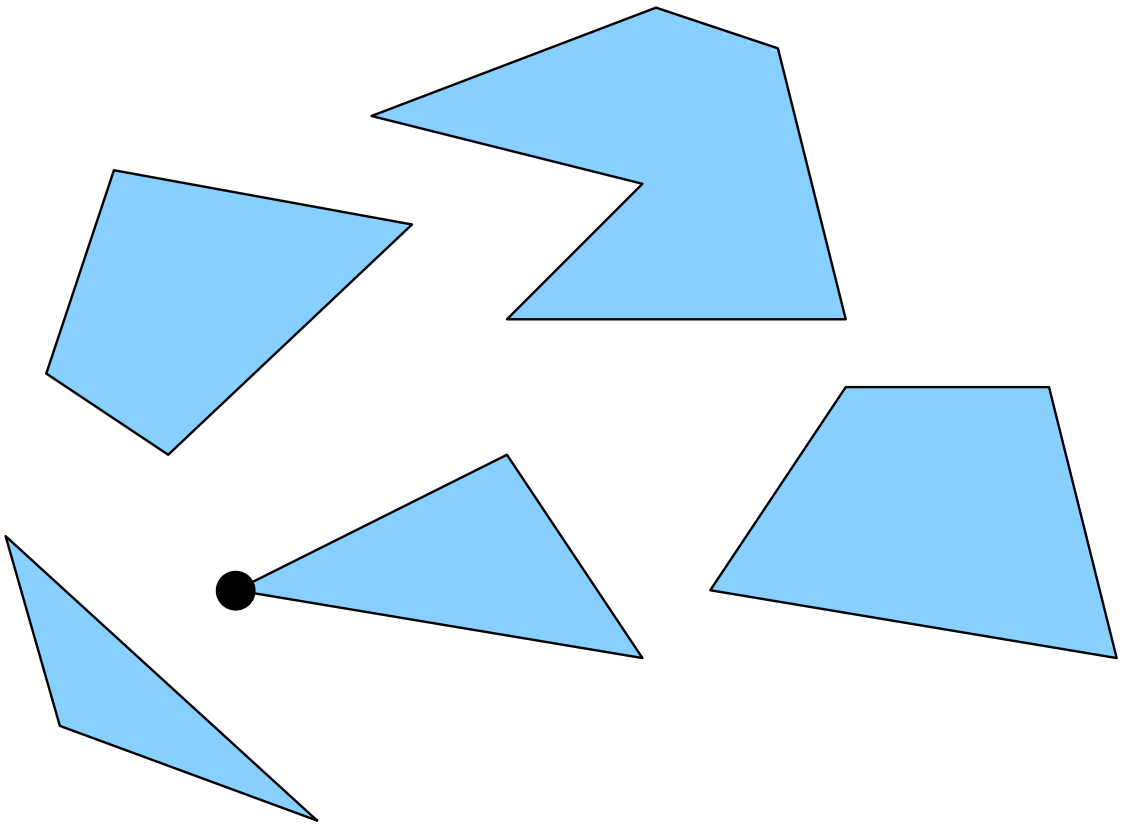
```
 1  if pv_i schneidet das Polygon von v_i lokal
       bei v_i
 2     then return  false

 3  if i = 1 or v_{i-1} ∉ pv_i
 4     then e ← linkeste Kante in T
 5          if e existiert nicht or
                v_i auf oder links von e
 6             then return  true
 7             else  return  false

    /* i ≠ 1 and v_{i-1} ∈ pv_i */
 8  if v_{i-1} ist nicht sichtbar
 9     then return  false

    /* v_{i-1} ist sichtbar */
10  Suche Kante e in T, die v_{i-1}v_i schneidet
11  if e existiert
12     then return  false
13     else  return  true
```

Beispiel



Analyse von SichtbareEcken

Sortieren:

Aufbau von $\mathcal{T}$:

$sichtbar(v_i)$:

Update von $\mathcal{T}$:

Satz

Der Sichtbarkeitsgraph einer Menge $\mathcal{H}$ von disjunkten polygonalen Hindernissen mit n Kanten kann in Zeit $O(n^2 \log n)$ berechnet werden.

4 Kürzeste Wege für polygonale Roboter

Vorlesung Bewegungsplanung:

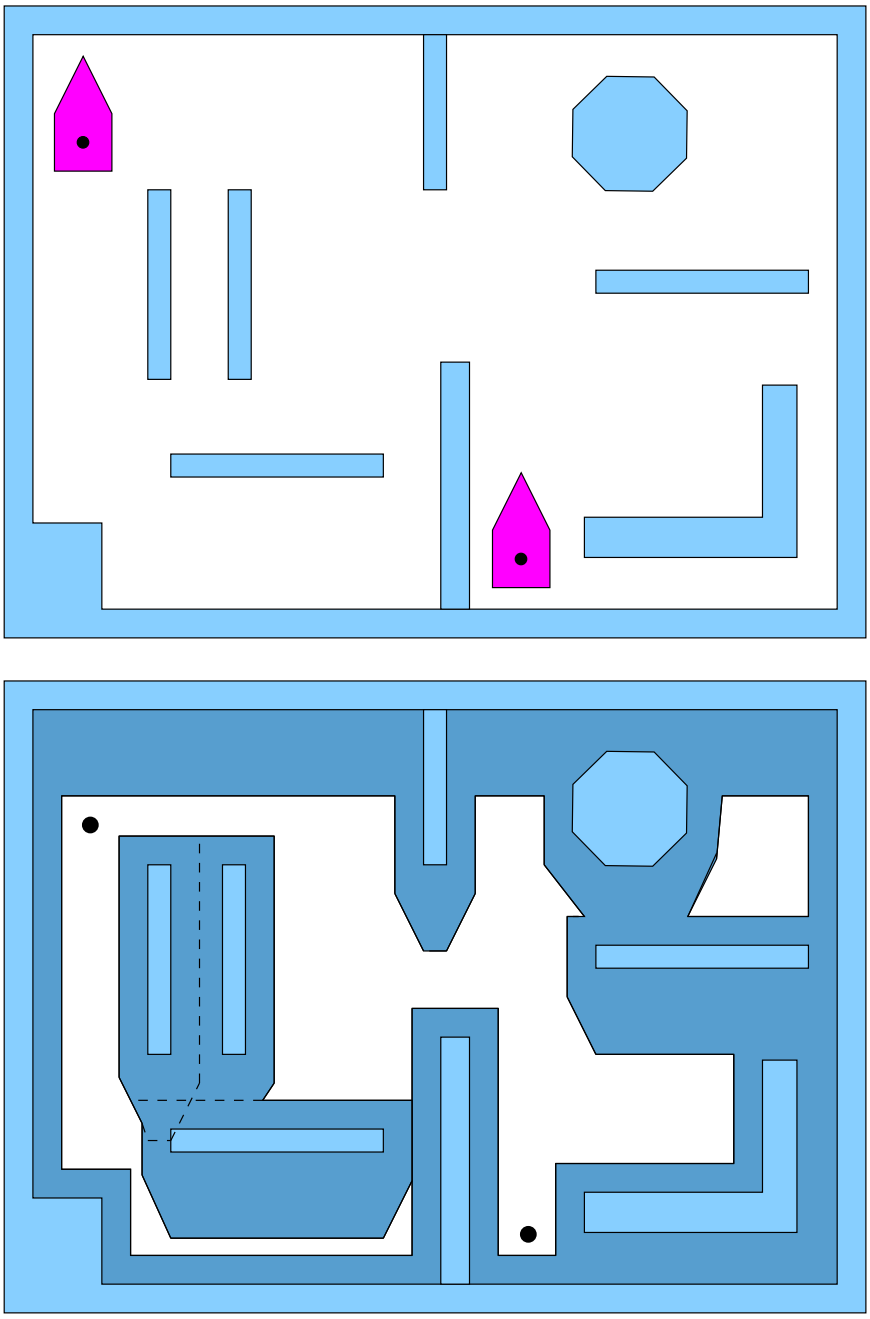
Translatorisches Bewegungsplanungproblem für polygonalen Roboter $\mathcal{R}$ zwischen Hindernissen $\{\mathcal{P}_1, \ldots, \mathcal{P}_t\}$

entspricht

Bewegungsplanungsproblem für Punktroboter zwischen Hindernissen

$$\{-\mathcal{R} \oplus \mathcal{P}_1, \ldots, -\mathcal{R} \oplus \mathcal{P}_t\}.$$

Kürzeste Wege für polygonale Roboter



Analyse

Berechnung von $\mathcal{C}_{free}$:

Komplexität von $\mathcal{C}_{free}$:

Berechnung des Sichtbarkeitsgraphen von $\mathcal{C}_{free}$:

Satz

Sei $\mathcal{R}$ ein konvexer Roboter mit konstant vielen Kanten, der translatorische Bewegungen zwischen einer Menge $\mathcal{H}$ von disjunkten Hindernissen mit insgesamt n Kanten ausführt.

Dann kann ein kürzester kollisionsfreier Weg von einer Startposition zu einer Endposition in Zeit $O(n^2 \log n)$ berechnet werden.