Vorlesung

Geometrische Algorithmen

Bewegungsplanung für Roboter

(Robot Motion Planning)

Sven Schuierer

Überblick

1. Problemstellung
2. Konfigurationsraum
3. Bewegungsplanung für einen Punktroboter
4. Minkowski Summen
5. Pseudo-Scheiben
6. Bewegungsplanung mit Translationen
7. Bewegungsplanung mit Translationen und Rotationen

1 Problemstellung

Zentrale Aufgabe für einen Roboter:

Planung einer Bewegung
(Bewegungsplanungsproblem)

Informationen über die Umgebung:

Beispiel: autonomer Roboter in einer Fabrik

- Grundriß für Wände und Maschinen
- Sensoren für Menschen

Arten von Robotern

Autonomer mobiler Roboter

Roboterarm

Problemstellung der Vorlesung

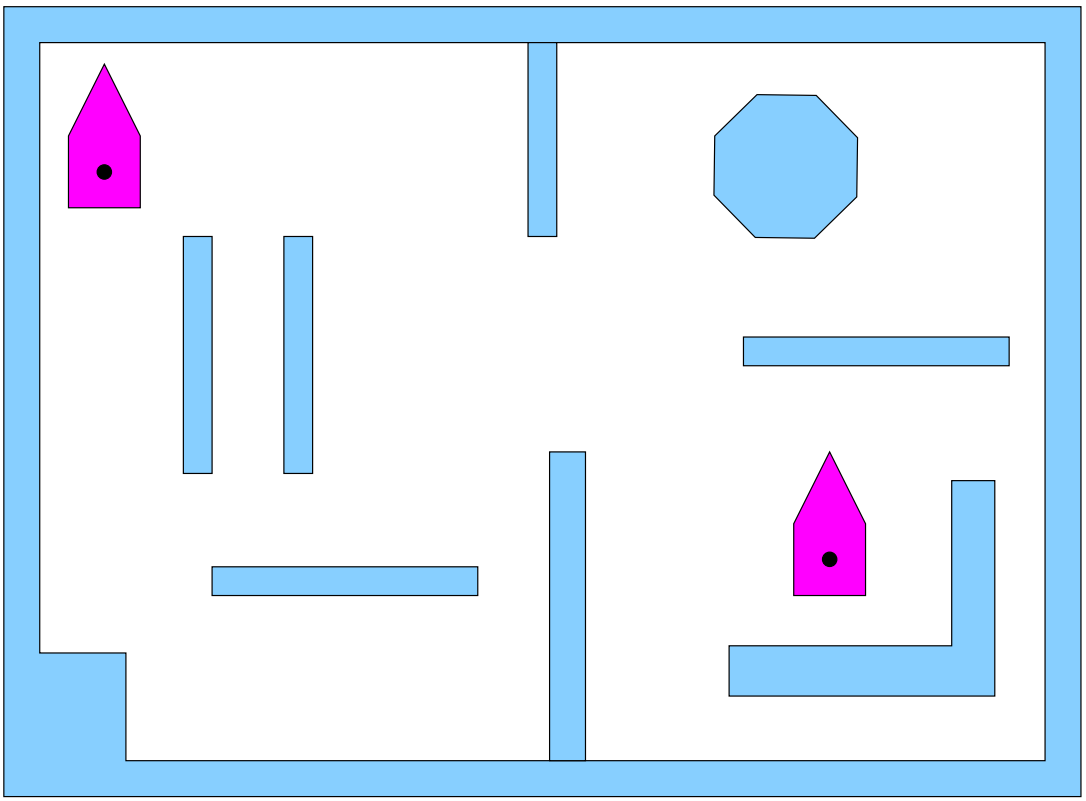
Vereinfachung: 2-dimensionales Bewegungsplanungsproblem

Umgebung (Arbeitsraum):

- planarer Bereich mit polygonalen Hindernissen $\mathcal{H}$
- statisch
- vollständig bekannt

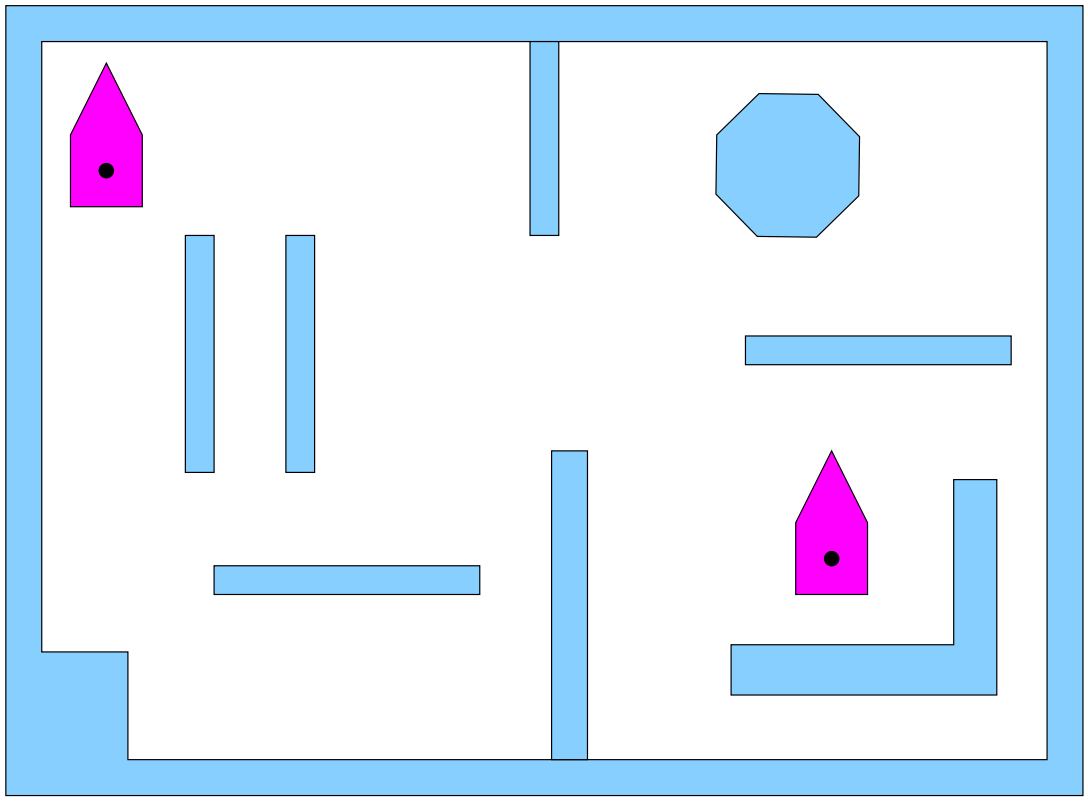
Roboter $\mathcal{R}$:

- einfaches Polygon
- in alle Richtungen beweglich



2 Arbeits- und Konfigurationsraum

Arbeitsraum: Menge $\mathcal{H}$ von Hindernissen $\{\mathcal{P}_1, \ldots, \mathcal{P}_t\}$

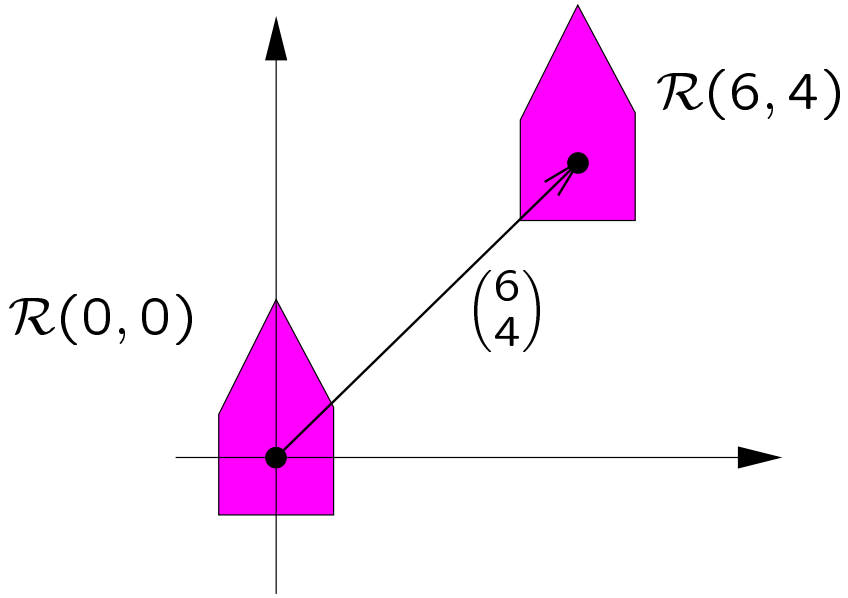


3 Konfigurationsraum

Konfiguration $\mathcal{R}(\vec{x})$: Spezifikation der Position des Roboters durch Parameter

Beispiel: Translatorischer Roboter $\mathcal{R}(x, y)$

$$\mathcal{R}(0,0) = (v_1, v_2, \ldots, v_r) \qquad \Rightarrow$$
$$\mathcal{R}(x,y) = (v_1 + (x,y), \ldots, v_r + (x,y))$$



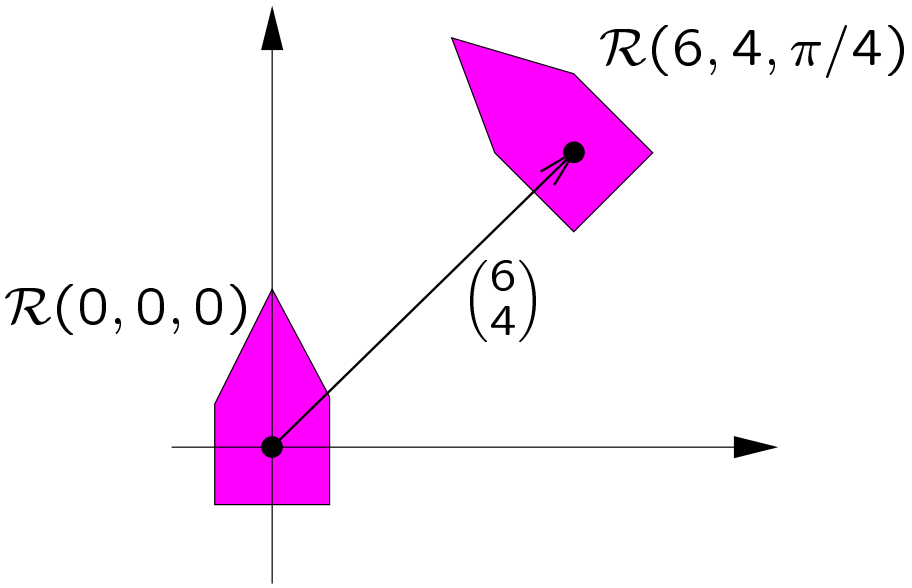
Alternative: Referenzpunkt

Ursprung $(0,0)$ des Koordinatensystems

Konfigurationen

Rotation: zusätzlicher Parameter φ

$\mathcal{R}(x, y, \varphi)$ = Roboter mit Referenzpunkt auf (x, y) und rotiert um φ Grad



Allgemein: **Freiheitsgrade**

3-dimensionaler Roboter:

Translation: drei Freiheitsgrade

Translation und Rotation: sechs Freiheitsgrade

Konfigurationsraum

Menge aller Parameter eines Roboters $\mathcal{R}$ heißt Konfigurationsraum $\mathcal{C}(\mathcal{R})$

$$p \in \mathcal{C}(\mathcal{R}) \longleftrightarrow \mathcal{R}(p)$$

Beispiele:

2-dimensionaler Roboter $\mathcal{R}$ mit Translation und Rotation:

$$(x, y, \varphi) \in \mathcal{C}(\mathcal{R}) = \mathbb{R}^2 \times [0, 2\pi) \mapsto \mathcal{R}(x, y, \varphi)$$

2-dimensionaler Roboter $\mathcal{R}$ mit Translation:

$$(x, y) \in \mathcal{C}(\mathcal{R}) = \mathbb{R}^2 \mapsto \mathcal{R}(x, y)$$

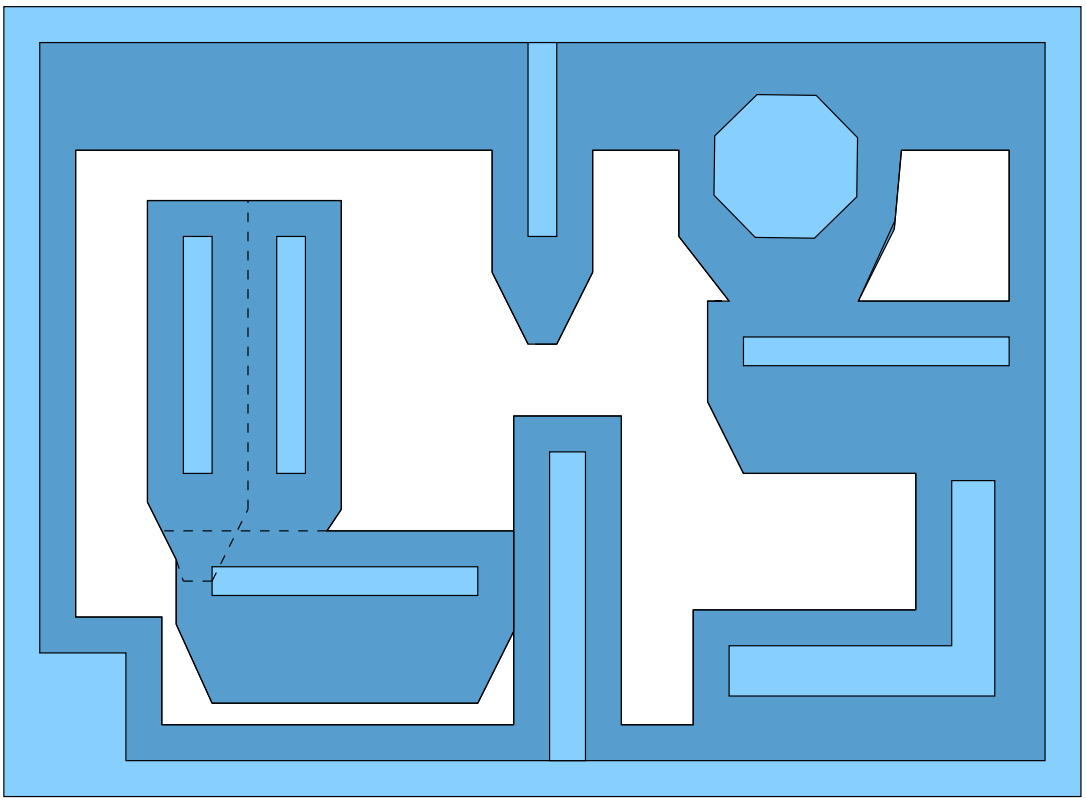
Verbotener und freier Raum

Verbotener Konfigurationsraum $\mathcal{C}_{forb}(\mathcal{R}, \mathcal{H})$:

$$\mathcal{C}_{forb}(\mathcal{R}, \mathcal{H}) = \{p \in \mathcal{C}(\mathcal{R}) \mid \text{Es gibt ein } \mathcal{P} \in \mathcal{H} : \mathcal{R}(p) \cap \mathcal{P} \neq \emptyset\}$$

Freier Konfigurationsraum $\mathcal{C}_{free}(\mathcal{R},\mathcal{H})$:

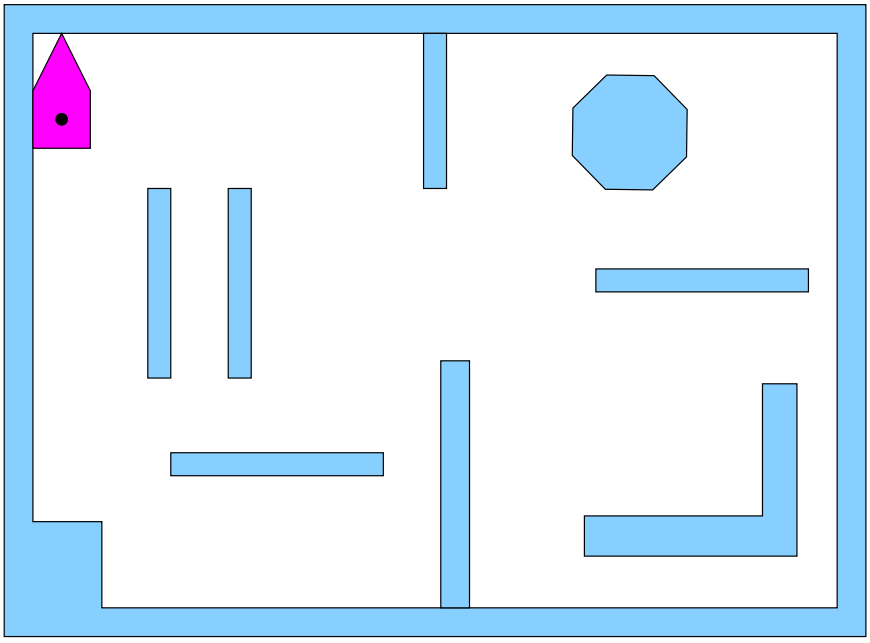
$$\mathcal{C}_{free}(\mathcal{R}, \mathcal{H}) = \mathcal{C}(\mathcal{R}) \setminus \mathcal{C}_{forb}(\mathcal{R}, \mathcal{H})$$

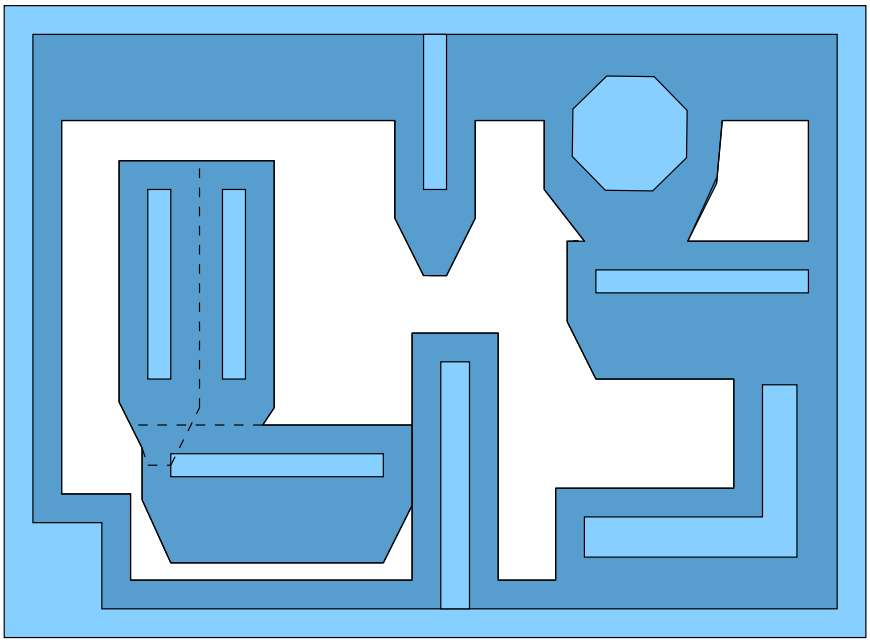


Arbeits- und Konfigurationsraum

Beobachtung

Jeder (kollisionsfreie) Weg W des Roboters im Arbeitsraum korrespondiert zu einem Weg $\mathcal{C}(W)$ im (freien) Konfigurationsraum und umgekehrt.





4 Punktroboter

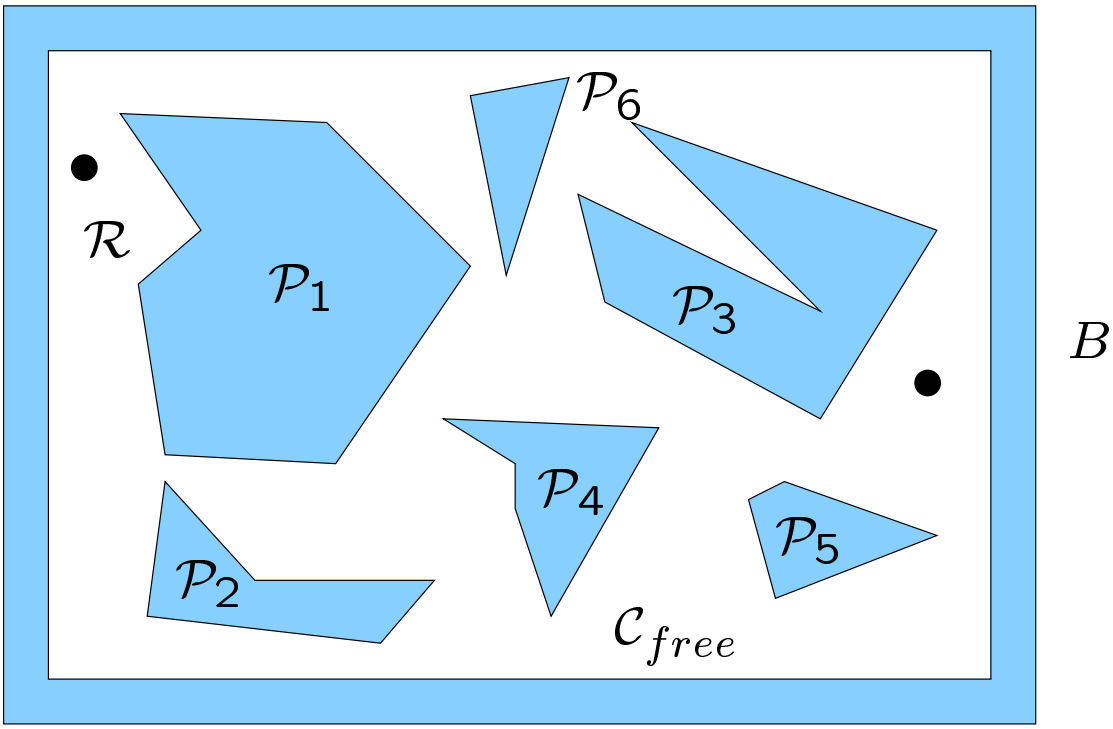
Annahmen:

- punktförmiger Roboter $\mathcal{R}$
- polygonale, disjunkte Hindernisse

  $$\mathcal{H} = \{\mathcal{P}_1, \ldots, \mathcal{P}_t\}$$

  mit insgesamt n Eckpunkten
- Rechteck B enthält die Hindernisse

$$\mathcal{C}_{free} = \mathcal{C}_{free}(\mathcal{R}, \mathcal{H}) = B \setminus \bigcup_{i=1}^{t} \mathcal{P}_i$$



Der freie Raum

Trapez-Zerlegung:

for all v Eckpunkt in $\mathcal{H}$ **do**

zeichne ein vertikales Liniensegment in v nach oben und eines nach unten

die Liniensegmente enden beim ersten Hindernis oder bei B

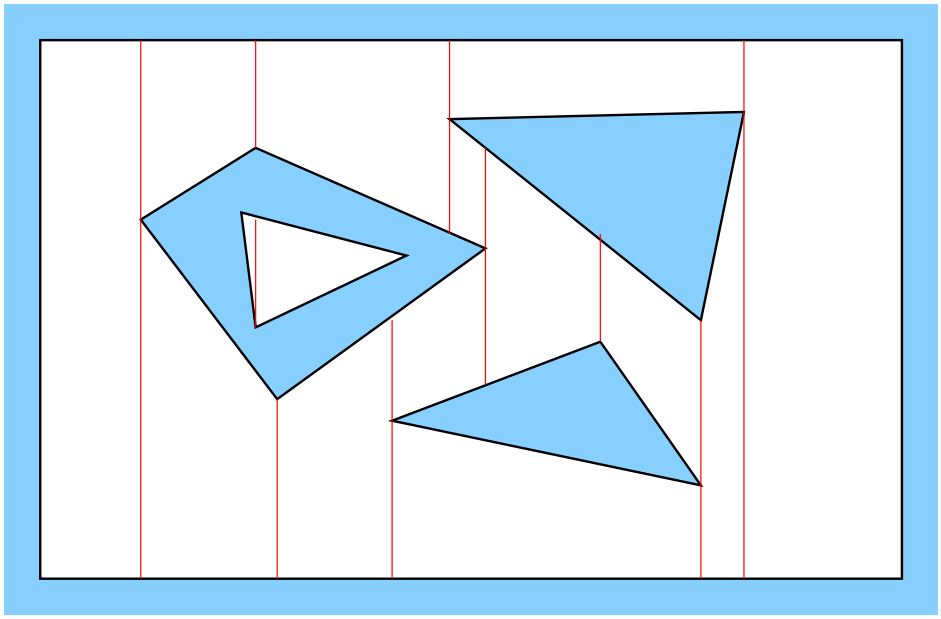
Algorithmus *BerechneFreienRaum($\mathcal{H}$)*

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

Output: Die vertikale Dekomposition von $\mathcal{C}_{free}(\mathcal{R},\mathcal{H})$ für einen Punktroboter $\mathcal{R}$

1. Sei E die Menge der Kanten in $\mathcal{H}$
2. Berechne die Trapez-Zerlegung $T(E)$ von E
3. Entferne die Trapeze im inneren der Hindernisse von $\mathcal{H}$
4. Gebe die entstehende Unterteilung $T(\mathcal{C}_{free})$ zurück

Der freie Raum



Lemma

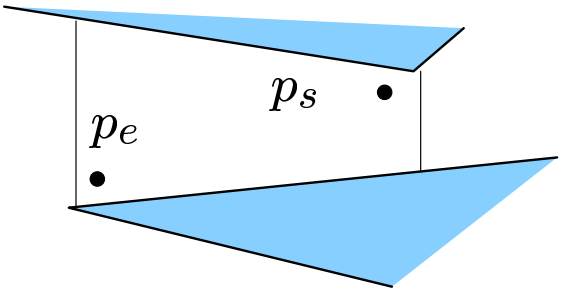
Die Trapez-Zerlegung des freien Konfigurationsraumes kann in $O(n \log n)$ erwarteter Zeit berechnet werden.

Berechnung eines Weges

Gegeben: p_s der Startpunkt und p_e der Zielpunkt

Gesucht: Ein kollisionsfreier Weg von p_s nach p_e

1. Fall: p_s und p_e gehören zum gleichen Trapez

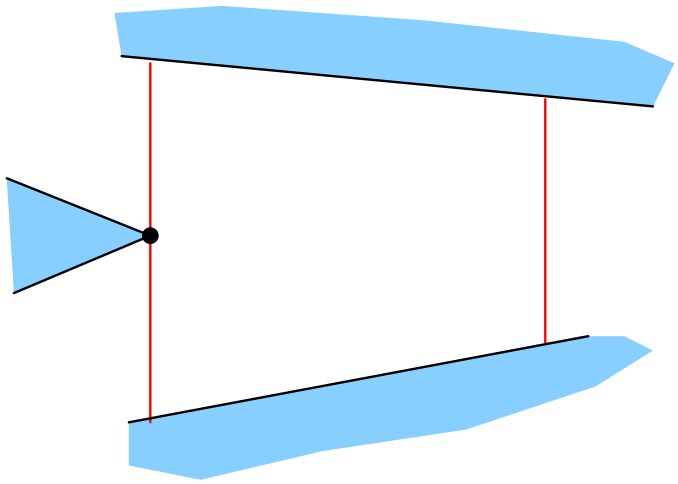


2. Fall: p_s und p_e gehören zu verschiedenen Trapezen $\Rightarrow$??

Die Straßenkarte (Road map)

Straßenkarte $\mathcal{G}_{road}$: planarer, in $\mathcal{C}_{free}$ eingebetter Graph

Knoten von $\mathcal{G}_{road}$: Trapez- und vertikale Kanten-Mittelpunkte

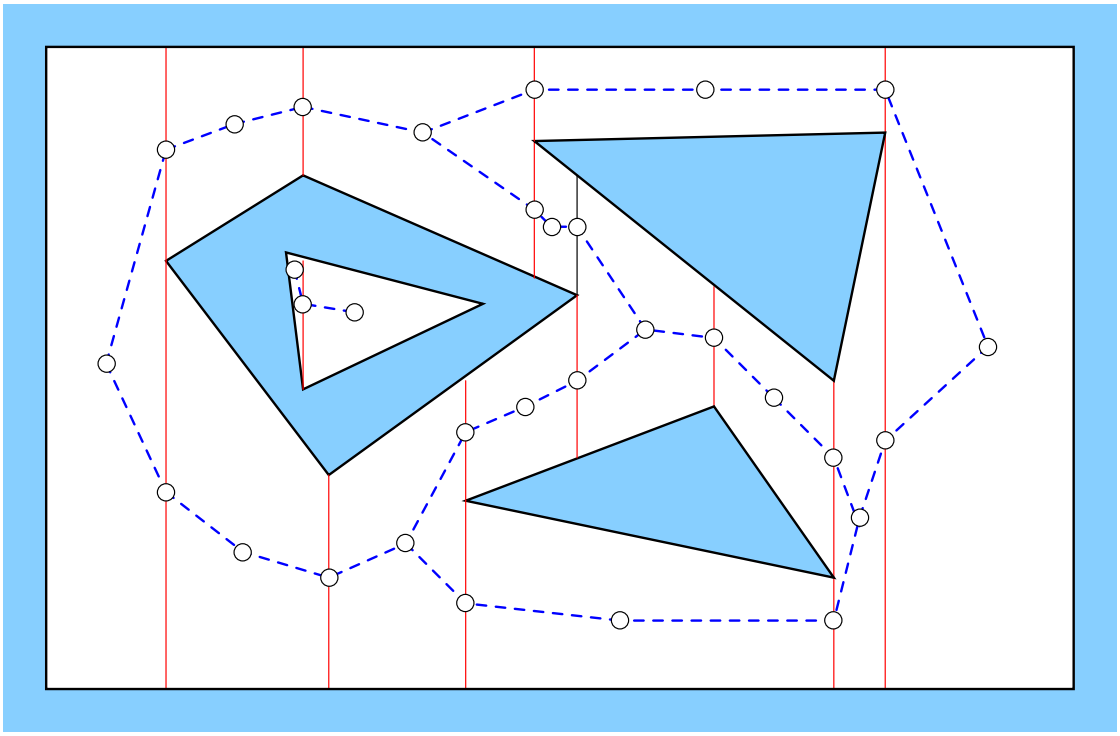


Kanten von $\mathcal{G}_{road}$:

for all Trapeze T **do**

füge Kanten von dem Mittelpunkt von T zu den Mittelpunkten der T begrenzenden vertikalen Kanten ein

Die Straßenkarte (Road map)



Zeit zur Konstruktion der Straßenkarte: $O(n)$

Algorithmus zur Wegeberechnung

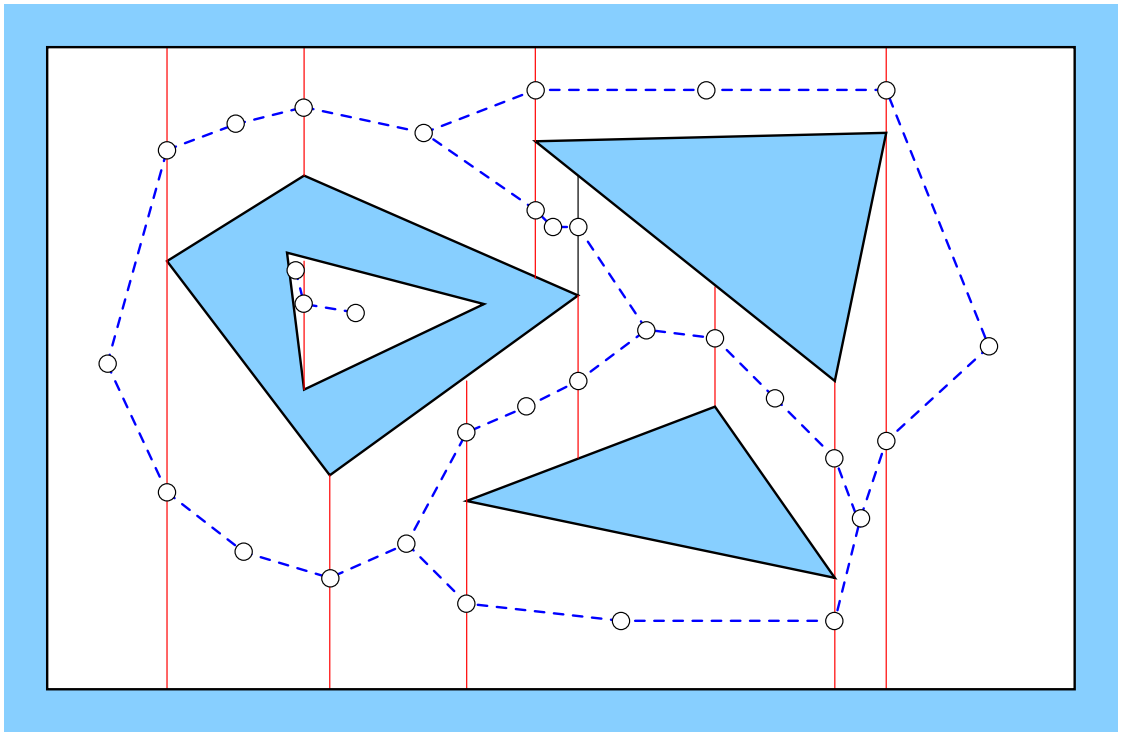
Algorithmus *BerechneWeg*

Input: vertikale Dekomposition $T(\mathcal{C}_{free})$, Straßenkarte $\mathcal{G}_{road}$, Start- und Zielpunkt p_s und p_e

Output: Ein Weg von p_s nach p_e, falls einer existiert; ansonsten eine Meldung, daß kein Weg existiert

```
1  Bestimme Trapeze Δ_s und Δ_e, die p_s und
   p_e enthalten
2  if Δ_s oder Δ_e existiert nicht
3     then melde, daß p_s oder p_e in C_forb
4     else seien v_s und v_e Mittelpunkte von
               Δ_s bzw. Δ_e
5          berechne Weg W von v_s nach v_e in
               G_road
6          if W existiert nicht
7             then melde, daß kein Weg
                        existiert
8             else return
                        W* = p_s v_s ∘ W ∘ v_e p_e
```

Beispiel



Korrektheit von BerechneWeg

Lemma

Algorithmus BerechneWeg gibt einen kollisionsfreien Weg zurück gdw es einen solchen Weg gibt.

Beweis:

$\Rightarrow$: W^* ist kollisionsfrei

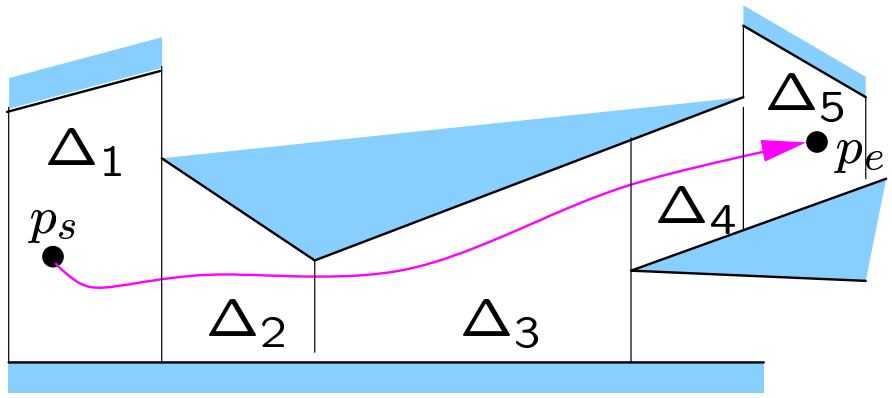
$\Leftarrow$: Sei W' kollisionfreier Weg von p_s nach p_e

W' schneidet eine Folge F von benachbarten Trapezen:

$$F = (\Delta_1, \Delta_2, \ldots, \Delta_k)$$

mit

$$\Delta_1 = \Delta_s \qquad \text{und} \qquad \Delta_k = \Delta_e$$

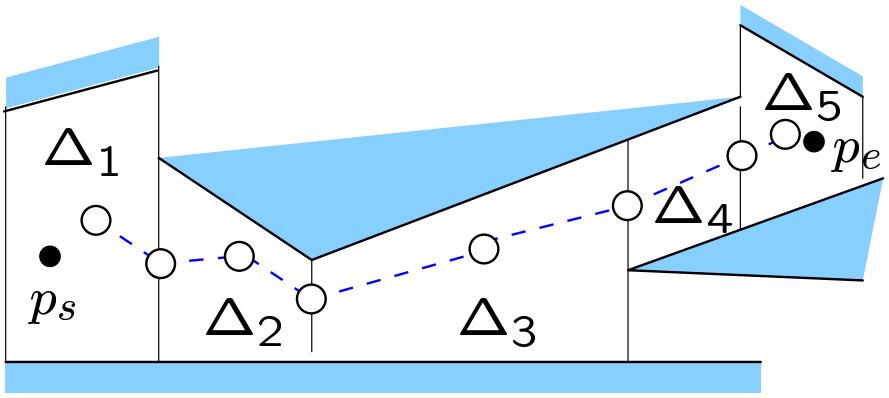


Korrektheit von BerechneWeg

Beweis: (Fortsetzung)

Sei v_i Mittelpunkt von Δ_i

Δ_i und Δ_{i+1} sind benachbart
$\Rightarrow$ Es gibt einen Weg von v_i nach v_{i+1} in $\mathcal{G}_{road}$
$\Rightarrow$ Es gibt einen Weg von v_1 nach v_k in $\mathcal{G}_{road}$
$\Rightarrow$ Der Algorithmus findet einen Weg von v_s nach v_e
$\Rightarrow$ Der Algorithmus findet einen Weg von p_s nach p_e



Zeitkomplexität von BerechneWeg: $O(n \log n)$ (erwartet)