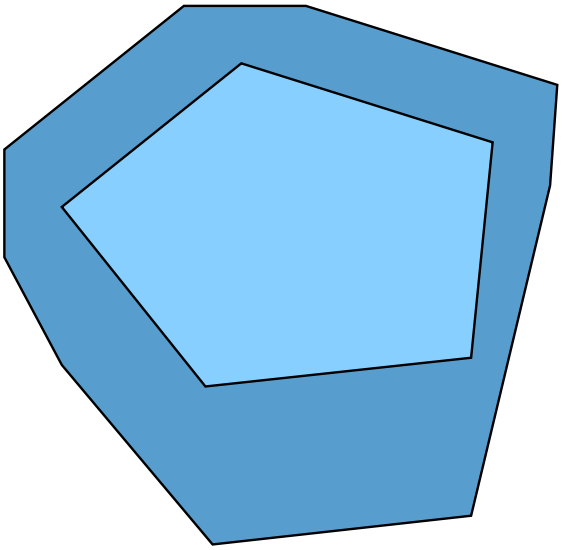
5 Minkowski Summen

Ziel: Berechnung des freien Konfigurationsraumes für polygonale Roboter

Definition

Konfigurationshindernis oder $\mathcal{C}$-Hindernis $\mathcal{C}(\mathcal{P})$ zu Hindernis $\mathcal{P}$:

$$\mathcal{C}(\mathcal{P}) = \{(x, y) \mid \mathcal{R}(x, y) \cap \mathcal{P} \neq \emptyset\}$$



Minkowski Summe

Definition

Seien $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathbb{R}^2$. Die Minkowski-Summe von $\mathcal{S}_1$ und $\mathcal{S}_2$ ist definiert als

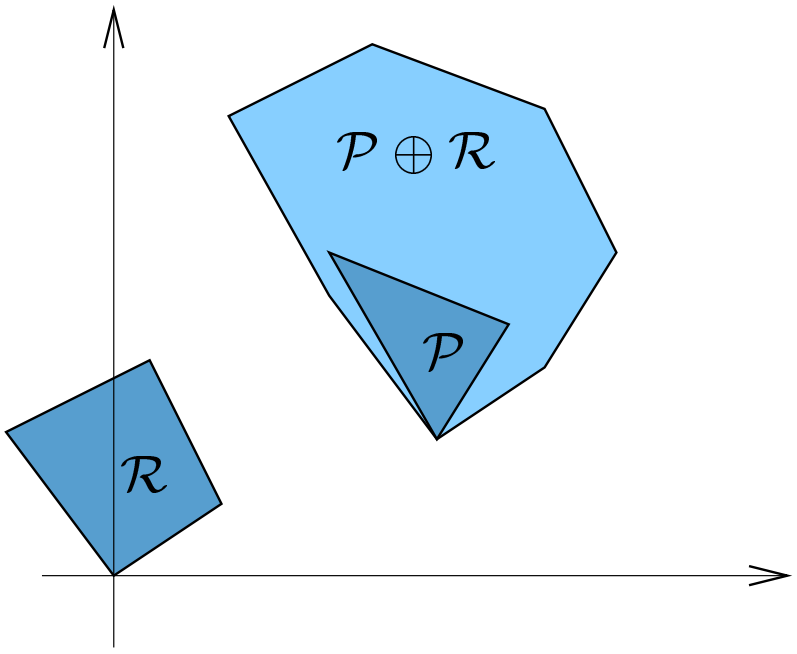
$$\mathcal{S}_1 \oplus \mathcal{S}_2 = \{p + q \mid p \in \mathcal{S}_1, q \in \mathcal{S}_2\}$$

wobei mit $p = (p_1, p_2)$ und $q = (q_1, q_2)$

$$p + q = (p_1 + q_1, p_2 + q_2)$$

Ferner sei

$$-p = (-p_1, -p_2) \qquad \text{und} \qquad -S = \{-p \mid p \in S\}$$



$\mathcal{C}$-Hindernisse

Satz

$$\mathcal{C}(\mathcal{P}) = \mathcal{P} \oplus (-\mathcal{R}(0,0))$$

Beweis:

"$\subseteq$": Sei $q \in \mathcal{C}(\mathcal{P})$ und $p \in \mathcal{P} \cap \mathcal{R}(q)$

$\Rightarrow p - q \in \mathcal{R}(0,0)$

$\Rightarrow -p + q \in \{(0,0)\} \oplus (-\mathcal{R}(0,0))$

$\Rightarrow q \in \{p\} \oplus (-\mathcal{R}(0,0)) \subseteq \mathcal{P} \oplus (-\mathcal{R}(0,0))$

"$\supseteq$": Sei $q \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$

$\Rightarrow q = p - r$, mit $p \in \mathcal{P}$, $r \in \mathcal{R}(0,0)$

$\Rightarrow p = q + r$

$\Rightarrow p \in P \cap \mathcal{R}(q)$

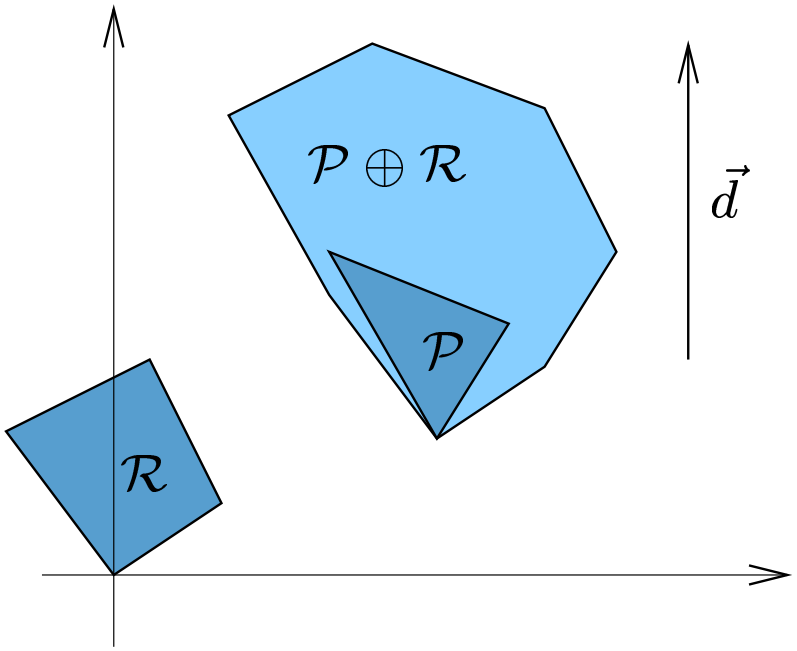
Extreme Punkte

Beobachtung

Sei

$$\mathcal{Q} = \mathcal{P} \oplus \mathcal{R},$$

dann ist ein in Richtung $\vec{d}$ extremer Punkt von $\mathcal{Q}$ die Summe von in Richtung $\vec{d}$ extremen Punkten von $\mathcal{P}$ und $\mathcal{R}$.



Komplexität von $\mathcal{P} \oplus \mathcal{R}$

Satz

Seien $\mathcal{P}$ und $\mathcal{R}$ konvex mit n bzw. mit m Kanten, dann ist

$$\mathcal{P} \oplus \mathcal{R}$$

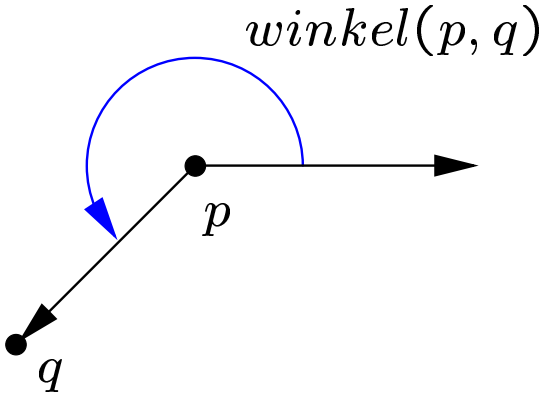
ein konvexes Polygon mit höchstens $n + m$ Kanten.

Beweis: Konvexität von $\mathcal{P} \oplus \mathcal{R}$ folgt aus der Definition

Sei e eine Kante von $\mathcal{P} \oplus \mathcal{R}$
$\Rightarrow$ e ist extrem in Richtung ihrer äußerer Normalen $\vec{u}_e$
$\Rightarrow$ e wird von in Richtung $\vec{u}_e$ extremen Punkten von $\mathcal{P}$ und $\mathcal{R}$ erzeugt
$\Rightarrow$ es gibt eine in Richtung $\vec{u}_e$ extreme Kante e' von $\mathcal{P}$ oder $\mathcal{R}$
$\Rightarrow$ weise e der Kante e' zu

Berechung der Minkowski-Summe

Definition



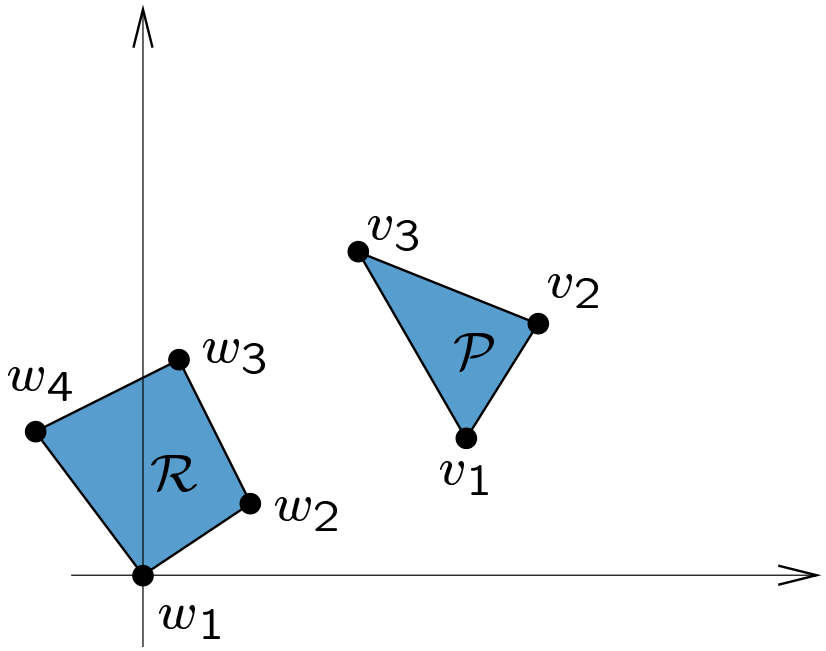
Algorithmus *Minkowski-Summe*

Input: Konvexe Polygone $\mathcal{P} = (v_1, \ldots, v_n)$ und $\mathcal{R} = (w_1, \ldots, w_m)$ mit v_1 und w_1 y-minimale Ecken

Output: Die Minkowski-Summe $\mathcal{P} \oplus \mathcal{R}$

```
 1  i ← 1; j ← 1
 2  v_{n+1} ← v_1; w_{m+1} ← w_1
 3  repeat
 4     Füge v_i + w_j als Ecke zu P ⊕ R hinzu
 5     if winkel(v_i, v_{i+1}) < winkel(w_j, w_{j+1})
 6        then i ← i + 1
 7     if winkel(v_i, v_{i+1}) > winkel(w_j, w_{j+1})
 8        then j ← j + 1
 9     if winkel(v_i, v_{i+1}) = winkel(w_j, w_{j+1})
10        then i ← i + 1, j ← j + 1
11  until i = n + 1 and j = m + 1
```

Beispiel



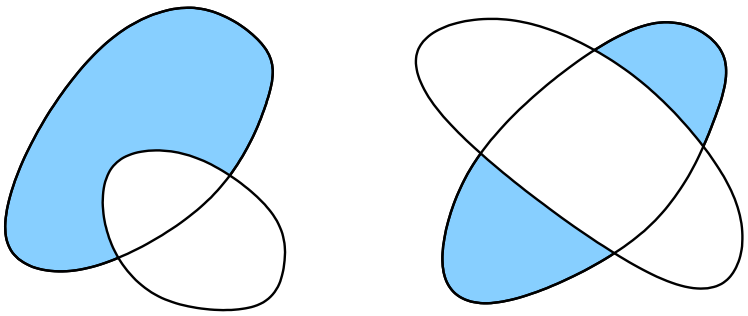
Satz

Die Minkowski-Summe zweier konvexer Polygone mit n bzw. m Kanten kann in Zeit $O(n+m)$ berechnet werden.

Pseudo-Scheiben

Definition

Zwei planare Objekte $\mathcal{O}_1$ und $\mathcal{O}_2$ heißen Pseudo-Scheiben, falls $\mathcal{O}_1 \setminus \mathcal{O}_2$ und $\mathcal{O}_2 \setminus \mathcal{O}_1$ zusammenhängend ist.



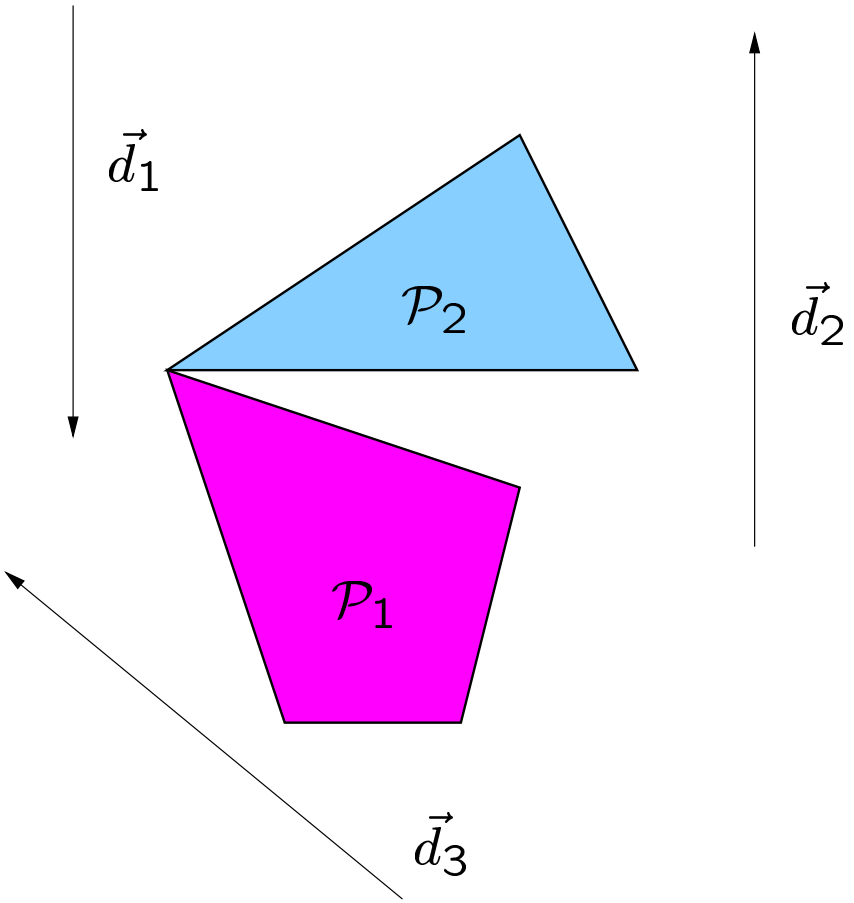
Beobachtung

Die Ränder zweier Pseudo-Scheiben haben höchstens zwei echte Kreuzungen.

Extreme Punkte

Definition

Seien $\mathcal{P}_1$ und $\mathcal{P}_2$ zwei Polygone mit disjunktem Innerem in der Ebene. $\mathcal{P}_1$ heißt extremer in Richtung $\vec{d}$ als $\mathcal{P}_2$, falls $\mathcal{P}_1$ Punkte besitzt, die weiter in Richtung $\vec{d}$ liegen als $\mathcal{P}_2$.

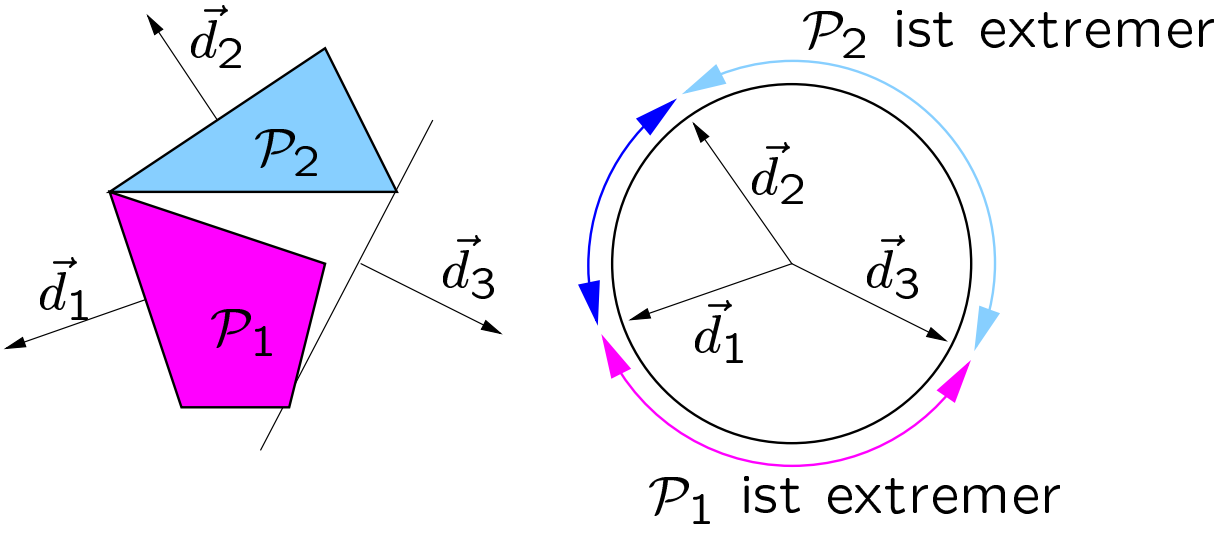


Extreme Punkte

Beobachtung

Seien $\mathcal{P}_1$ und $\mathcal{P}_2$ zwei Polygone in der Ebene mit disjunktem Inneren und $\vec{d}_1$ und $\vec{d}_2$ zwei Richtungen, so daß $\mathcal{P}_1$ extremer als $\mathcal{P}_2$ in Richtung $\vec{d}_1$ und $\vec{d}_2$ ist.

Dann ist $\mathcal{P}_1$ entweder in allen Richtungen von $\vec{d}_1$ nach $\vec{d}_2$ oder in allen Richtungen von $\vec{d}_2$ nach $\vec{d}_1$ extremer als $\mathcal{P}_2$.



Minkowski-Summen als Pseudo-Scheiben

Satz

Seien $\mathcal{P}_1$ und $\mathcal{P}_2$ zwei konvexe Polygone mit disjunktem Innerem und $\mathcal{R}$ ein weiteres konvexes Polygon. Dann sind $\mathcal{C}_1 = \mathcal{P}_1 \oplus \mathcal{R}$ und $\mathcal{C}_2 = \mathcal{P}_2 \oplus \mathcal{R}$ Pseudo-Scheiben.

Beweis:

Zu zeigen: $\mathcal{C}_1 \setminus \mathcal{C}_2$ und $\mathcal{C}_2 \setminus \mathcal{C}_1$ sind zusammenhängend

Annahme: $\mathcal{C}_1 \setminus \mathcal{C}_2$ besteht aus mindestens zwei Zus.hangskomponenten $\mathcal{Z}_1$ und $\mathcal{Z}_2$

$\mathcal{C}_1$ und $\mathcal{C}_2$ sind konvex
$\Rightarrow$ es gibt $\vec{d}_1 \in \mathcal{Z}_1$ und $\vec{d}_2 \in \mathcal{Z}_2$ mit $\mathcal{C}_1$ ist extremer als $\mathcal{C}_2$ in Richtung $\vec{d}_1$ und $\vec{d}_2$
$\Rightarrow$ $\mathcal{P}_1$ ist extremer als $\mathcal{P}_2$ in Richtung $\vec{d}_1$ und $\vec{d}_2$

$\mathcal{P}_1$ und $\mathcal{P}_2$ haben disjunkte Innere
$\Rightarrow$ $\mathcal{P}_1$ ist extremer als $\mathcal{P}_2$ für alle Richtungen von $\vec{d}_1$ bis $\vec{d}_2$ (oder $\vec{d}_2$ bis $\vec{d}_1$)
$\Rightarrow$ $\mathcal{C}_1$ ist extemer als $\mathcal{C}_2$ für alle Richtungen von $\vec{d}_1$ bis $\vec{d}_2$ **Widerspruch!**

Vereinigung von Pseudo-Scheiben

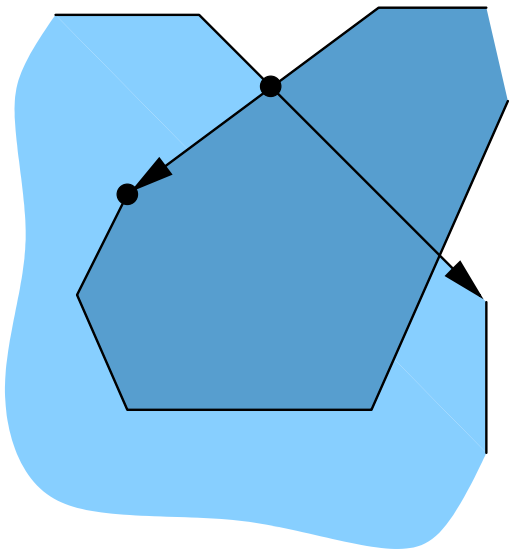
Satz

Sei $\mathcal{S}$ eine Menge von polygonalen Pseudo-Scheiben mit insgesamt n Kanten. Dann ist die Komplexität ihrer Vereinigung $\mathcal{V}$ höchstens $2n$.

Beweis: $\mathcal{V}$ enthält zwei Arten von Ecken
1. Ecken der Polygone in $\mathcal{S}$ und
2. Kantenschnittpunkte von Polygonen in $\mathcal{S}$

zu 2.: Betrachte Kantenschnittpunkt $v = e \cap e'$
Endpunkt v' von e oder e' liegt im Inneren von $\mathcal{V}$
$\Rightarrow$ Weise v v' zu

v' hat zwei inzidente Kanten
$\Rightarrow$ höchstens zwei zugewiesene Kantenschnittpunkte



Nicht-konvexe Polygone

Es gilt:

$\mathcal{S}_1 \oplus (\mathcal{S}_2 \cup \mathcal{S}_3) \;=$

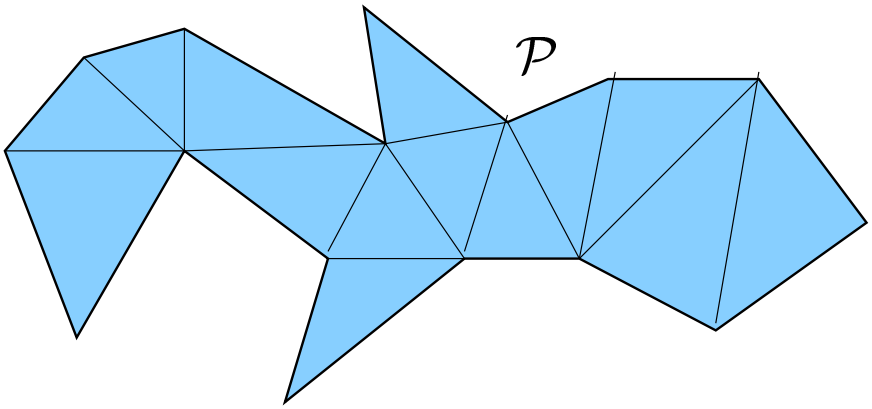
$\mathcal{P}$ **nicht konvex:** $\Rightarrow$?

Nicht-konvexe Polygone

$$\mathcal{S}_1 \oplus (\mathcal{S}_2 \cup \mathcal{S}_3) \;=\; (\mathcal{S}_1 \oplus \mathcal{S}_2) \cup (\mathcal{S}_1 \oplus \mathcal{S}_3)$$

$\mathcal{P}$ nicht konvex:

1. trianguliere $\mathcal{P} \Rightarrow$ Dreiecke $T_1, \ldots, T_{n-2}$

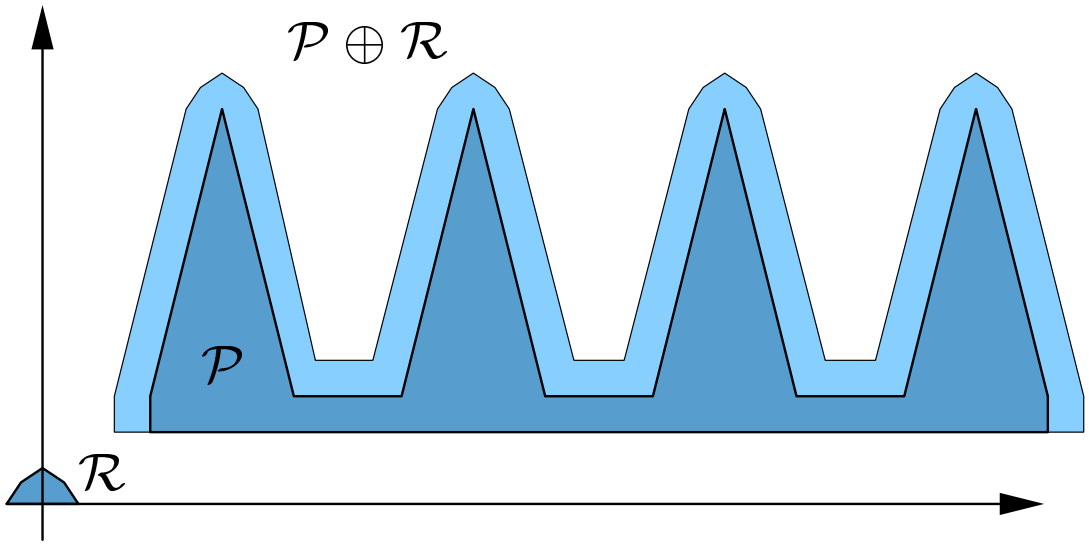


2. berechne

$$\mathcal{P} \oplus \mathcal{R} = \bigcup_{i=1}^{n-2} T_i \oplus \mathcal{R}$$

Komplexität: $O(nm)$

Nicht-konvexe Polygone



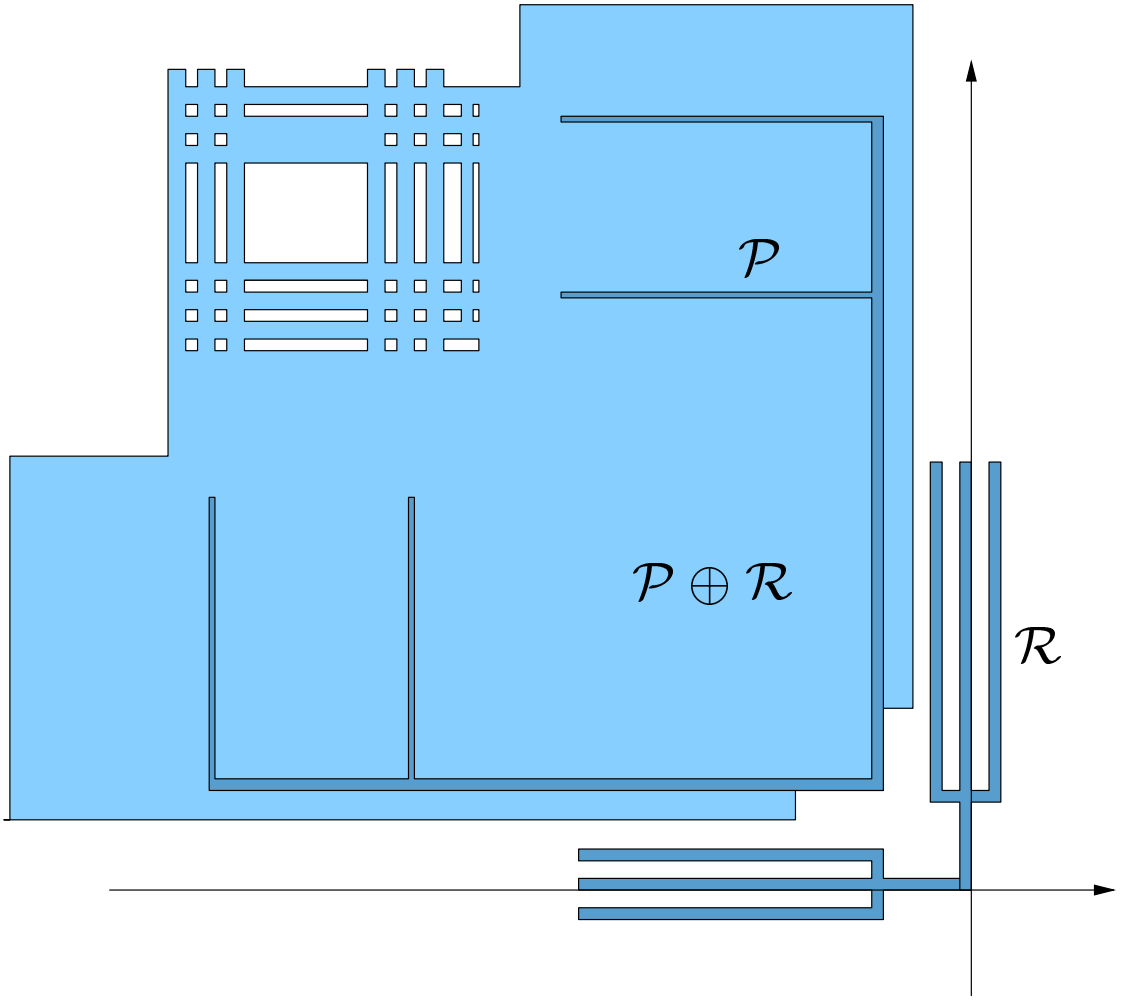
Nicht-konvexe Polygone

$\mathcal{P}$ und $\mathcal{R}$ nicht konvex:

1. trianguliere $\mathcal{P} \Rightarrow$ Dreiecke $T_1, \ldots, T_{n-2}$
2. trianguliere auch $\mathcal{R} \Rightarrow$ Dreiecke $\mathcal{U}_1, \ldots, \mathcal{U}_{m-2}$
3. berechne

$$\mathcal{P} \oplus \mathcal{R} = \bigcup_{i=1}^{n-2} \bigcup_{j=1}^{m-2} T_i \oplus \mathcal{U}_j$$

Komplexität: $O(n^2 m^2)$



6 Bewegungsplanung mit Translationen

$\mathcal{C}$-Hindernisse:

$$\mathcal{P}_i \oplus (-\mathcal{R}(0,0))$$

Satz

Sei $\mathcal{R}$ ein Roboter mit konstant vielen Kanten und $\mathcal{H}$ eine Menge von Hindernissen mit n Kanten. Dann ist die Komplexität des freien Konfigurationsraumes für Translationen $O(n)$.

Seien $T_1, \ldots, T_n$ die Dreiecke einer Triangulation der Hindernisse

Wir berechnen

$$\mathcal{C}_{forb} = \bigcup_{i=1}^{n} \mathcal{C}_i = \bigcup_{i=1}^{n} T_i \oplus (-\mathcal{R}(0,0))$$

Berechung von $\mathcal{C}_{forb}$

Algorithmus *VerbotenerRaum*

Input: Eine Menge $\mathcal{H} = \{T_1, \ldots, T_n\}$ von Dreiecken und ein Roboter $\mathcal{R}$

Output: $\mathcal{C}_{forb} = \bigcup_{i=1}^{n} \mathcal{C}_i = \bigcup_{i=1}^{n} T_i \oplus (-\mathcal{R}(0,0))$

```
1 if n = 1
2    then return T_1 ⊕ (−R(0,0))
3    else C^1_forb = VerbotenerRaum(T_1, ..., T_⌊n/2⌋)
4         C^2_forb = VerbotenerRaum(T_⌊n/2⌋+1, ..., T_n)
5         berechne C_forb = C^1_forb ∪ C^2_forb
6         return C_forb
```

Analyse VerbotenerRaum

Triangulierung: $|\mathcal{P}_i| = m_i$

Berechnung der $\mathcal{C}$-Hindernisse:

Berechnung von $\mathcal{C}^1_{forb} \cup \mathcal{C}^2_{forb}$:

$|\mathcal{C}^1_{forb}| = n_1,\ |\mathcal{C}^2_{forb}| = n_2,\ |\mathcal{C}_{forb}| = k$

Rekursion:

Zusammenfassung

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.

Man kann $\mathcal{H}$ in $O(n \log^2 n)$ erwarteter Zeit vorverarbeiten, so daß für jede Start- und Zielposition ein kollisionsfreier Weg für $\mathcal{R}$, sofern er existiert, in $O(n)$ Zeit berechnet werden kann.

7 Bewegungsplanung mit Translationen und Rotationen

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

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

- drei Freiheitsgrade: zwei translatorische und einen für die Rotation

Konfigurationsraum:

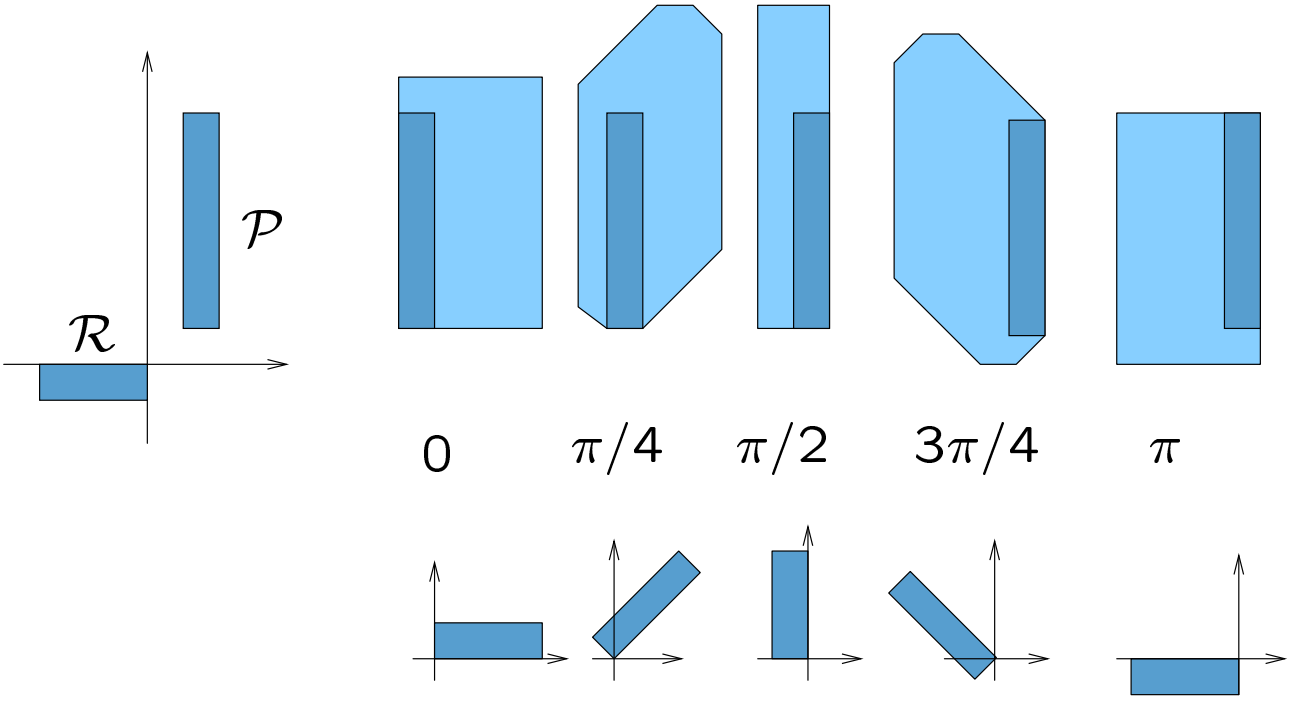
$$(x, y, \varphi) \in \mathcal{C}(\mathcal{R}) = \mathbb{R}^2 \times [0, 2\pi) \mapsto \mathcal{R}(x, y, \varphi)$$

Konfigurationshindernisse:

$$\mathcal{C}(\mathcal{P}_i) = \{(x, y, \varphi) \mid \mathcal{R}(x, y, \varphi) \cap \mathcal{P}_i \neq \emptyset\}$$

Konfigurationshindernisse

Betrachte Schnitte mit den Ebenen $h : \varphi = \varphi_0$



Minkowski-Summe

$$\mathcal{P}_i + (-\mathcal{R}(0,0,\varphi))$$

ändert sich stetig in φ.

Ein Algorithmus

Definition

Der Schnitt des Konfigurationsraums von $\mathcal{R}$ mit einer Ebene $h : \varphi = \varphi_0$ heißt Scheibe $\mathcal{S}_{\varphi_0}$ des Konfigurationsraums.

Idee:

- Zerteile Konfigurationsraum in Scheiben
- Zerlege die Bewegung des Roboters in Translationen innerhalb einer Scheibe und
- Rotationen zwischen Scheiben

$$m = \text{Anzahl der Scheiben}$$

Scheiben werden für Winkel φ_i berechnet:

$$\varphi_i = i \times \frac{2\pi}{m}$$

Ein Algorithmus

Algorithmus

Input: ein konvexer polygonaler Roboter $\mathcal{R}$ und eine Menge von Hindernisse $\mathcal{H}$

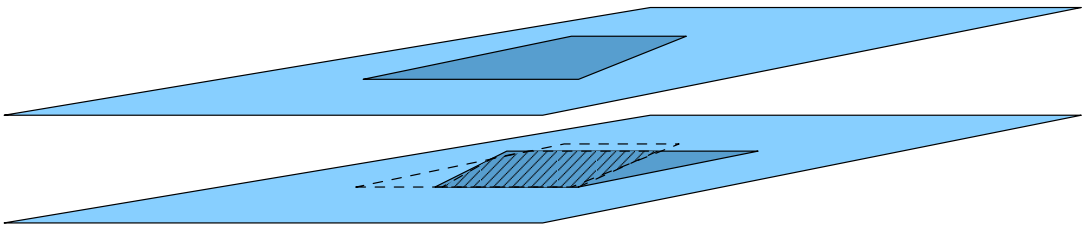
Output: eine Straßenkarte $\mathcal{G}_{road}$ für $\mathcal{C}(\mathcal{R})$

1 $T_{-1} \leftarrow \emptyset$; $\mathcal{G}_{-1} \leftarrow (\emptyset, \emptyset)$; $\mathcal{G}_{road} \leftarrow \emptyset$
2 **for** $i \leftarrow 0$ **to** $m - 1$ **do**
3 berechne Scheibe $\mathcal{S}_{\varphi_i}$ von $\mathcal{C}(\mathcal{R}, \mathcal{H})$
4 berechne Trapezzerlegung T_i von $\mathcal{S}_{\varphi_i}$
5 berechne Straßenkarte $\mathcal{G}_i$ von T_i
6 verknüpfe $\mathcal{G}_i$ mit $\mathcal{G}_{i-1}$ und füge $\mathcal{G}_i$ und die Verknüpfungen $\mathcal{G}_{road}$ hinzu
7 verknüpfe $\mathcal{G}_0$ mit $\mathcal{G}_{m-1}$ und füge die Verknüfungen zu $\mathcal{G}_{road}$ hinzu
8 **return** $\mathcal{G}_{road}$

Verknüpfung zweier Straßenkarten

Algorithmus *Verknüpfung von $\mathcal{G}_i$ mit $\mathcal{G}_{i-1}$*

1 Berechne die Überlagerung von T_{i-1} und T_i
2 Sei $\mathcal{U}$ die Menge der Trapezpaare
$(\Delta_1, \Delta_2) \in T_{i-1} \times T_i$ mit $\Delta_1 \cap \Delta_2 \neq \emptyset$
3 **for all** $(\Delta_1, \Delta_2) \in \mathcal{U}$ **do**
4 Sei (x, y) der Mittelpunkt von $\Delta_1 \cap \Delta_2$
5 Füge $v_1 = (x, y, \varphi_{i-1})$ als Knoten zu $\mathcal{G}_{i-1}$ hinzu und verbinde v_1 mit dem Mittelpunkt von Δ_1
6 Füge $v_2 = (x, y, \varphi_i)$ als Knoten zu $\mathcal{G}_i$ hinzu und verbinde v_2 mit dem Mittelpunkt von Δ_2
7 Verbinde v_1 und v_2 durch eine Kante



Berechung eines Weges

Algorithmus

Input: Startkonfiguration $\mathcal{R}(x_s, y_s, \varphi_s)$ und Zielkonfiguartion $\mathcal{R}(x_z, y_z, \varphi_z)$, Straßenkarte $\mathcal{G}_{road}$

Output: Ein Weg von (x_s, y_s, φ_s) nach (x_z, y_z, φ_z) oder eine Meldung, daß kein Weg existiert

1 $i_s \leftarrow round(m * \varphi_s)$; $i_z \leftarrow round(m * \varphi_z)$
2 Berechne Trapeze $\Delta_s \in T_{i_s}$ und $\Delta_z \in T_{i_z}$, die $(x_s, y_s, \varphi_{i_s})$ und $(x_z, y_z, \varphi_{i_z})$ enthalten
3 **if** Δ_s oder Δ_z existiert nicht
4 **then** melde es gibt keinen Weg; **return**
5 Sei v_s der Mittelpunkt von Δ_s und v_z der Mittelpunkt von Δ_z
6 Berechne einen Weg W_{road} in $\mathcal{G}_{road}$ von v_s nach v_z
7 **if** W_{road} existiert nicht
8 **then** melde es gibt keinen Weg; **return**
9 Gebe Weg von (x_s, y_s, φ_s) über $(x_s, y_s, \varphi_{i_s})$, v_s, W_{road}, v_z, $(x_z, y_z, \varphi_{i_z})$ nach (x_z, y_z, φ_z) zurück

Probleme

Algorithmus ist nicht korrekt

Beispiel:

$(x_s, y_s, \varphi_s) \in \mathcal{C}_{free}(\mathcal{R}, \mathcal{H})$, aber
$(x_s, y_s, \varphi_{i_s}) \notin \mathcal{C}_{free}(\mathcal{R}, \mathcal{H})$

Schlimmer:

Wege sind nicht kollisionsfrei

- Translationen innerhalb einer Scheibe sind kollisionsfrei
- Rotationen sind zwischen freien Konfigurationen

Lösung:

1. Erhöhung von m
2. Verwende vergrößerten (rotierten) Roboter