Offline Bewegungsplanung: Reine Translation

Elmar Langetepe
University of Bonn

Folgerungen!! Theorem 2.16

- Konvexer Roboter R mit $|R| = m$, polygonale Szene, n Kanten
- Komplexität von C_{verb} in $O(mn)$, Beweis!!
- Trianguliere alle P_j, $T_1, T_2, \ldots T_l$ mit $O(n)$ Kanten
- $CT_i = T_i \oplus -R(0,0)$ Familie von Pseudokreisen Lem. 2.12, $O((m+3)n)$ Kanten Lem. 2.15

$$C_{\text{verb}} = \underbrace{\bigcup_i T_i \oplus -R(0,0)}_{\text{Komplexität } O(mn)}.$$

Benutze Theorem 2.13

Folgerungen!! Theorem 2.16

- Algorithmus 2.2: Berechnung von C_{verb}
  - Unterteile Hindernisse in gleichgroße Teilmengen von Dreiecken CP_1 und CP_2
  - Berechne rek. $C_{\mathrm{verb}}(CP_1)$ und $C_{\mathrm{verb}}(CP_2)$, je $O(mn)$ Kanten
  - Berechne Vereinigung $C_{\mathrm{verb}}(CP_1) \cup C_{\mathrm{verb}}(CP_2)$ in Zeit $O(mn \log(mn))$ mit Sweep (Alg. Geom.)

Laufzeit insgesamt: $O(mn \log^2 mn)$

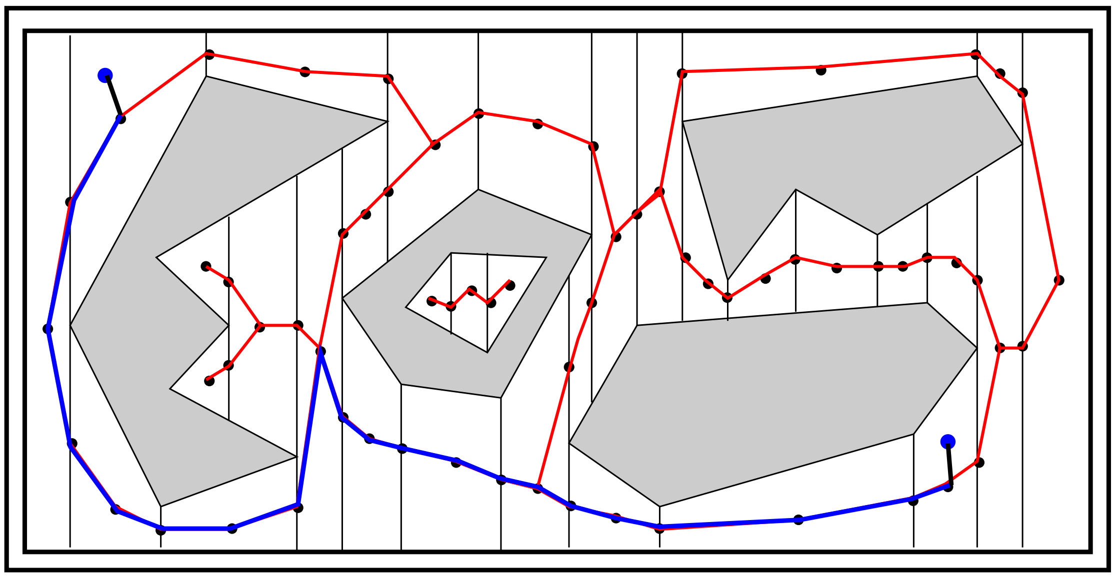
Laufzeitanalyse: Theorem 2.16

Arrangement mit $O(nm)$ Kanten!!

$$\begin{aligned}
T(m,n) &\leq 2T\left(m,\frac{n}{2}\right) \text{ (Rek.)} + C \times mn \log mn \text{ (Merge)} \\
&\leq 2\left(2T\left(m,\frac{n}{4}\right) + C\frac{mn}{2} \log \frac{mn}{2}\right) + C \times mn \log mn \\
&\vdots \quad \vdots \\
&\leq nT(m,1) + Cmn \sum_{i=0}^{\log n} \log \frac{mn}{2^i} \\
&\in O(mn \log^2 mn)
\end{aligned}$$

C_{frei} **verwenden! Roadmap!!**

- Zerlegung in Trapezen (Seidel)
- Benachbarte Trapeze verbinden (Schwerpunkt/Mittelpunkt)
- Zusammenhangsgraph, Roadmap
- Lokalisation s und t und verbinden (BFS)



Algorithmus 2.3: C_{frei} gegeben!

Vorbereitung:

- Trapezzerlegung C_{frei}/Point–Location Struktur, (randomisiert!!) $O(mn \log(mn))$
- Roadmap-Konstruktion: Verbindung adjazenter Trapeze, $O(mn)$

Query:

- Lokalisiere s und t Trapeze, $O(\log(mn))$
- Finde Weg der Trapeze mit BFS, $O(mn)$

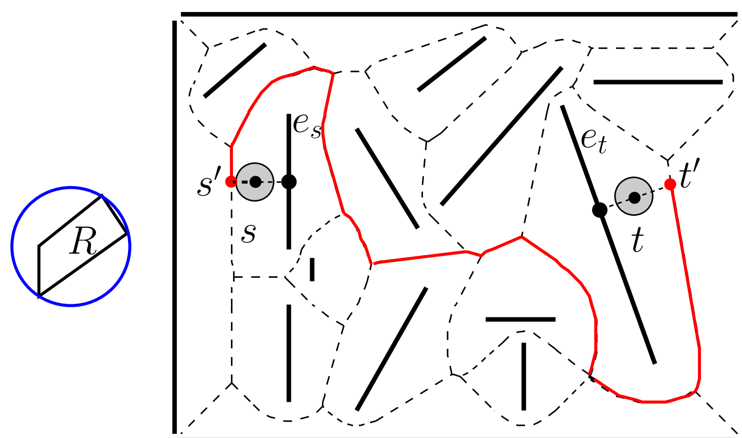
Roadmap!! Einfach aber effizient!! Probabilistisch!

Ergebnis: Theorem 2.17

Translationsbewegungen eines konvexen, polygonalen Roboters mit m Ecken in einer Umgebung mit polygonalen Hindernissen mit insgesamt n Ecken können nach $O(mn \log^2(mn))$ (randomisierter) Vorbereitungszeit in Zeit $O(mn)$ geplant werden.

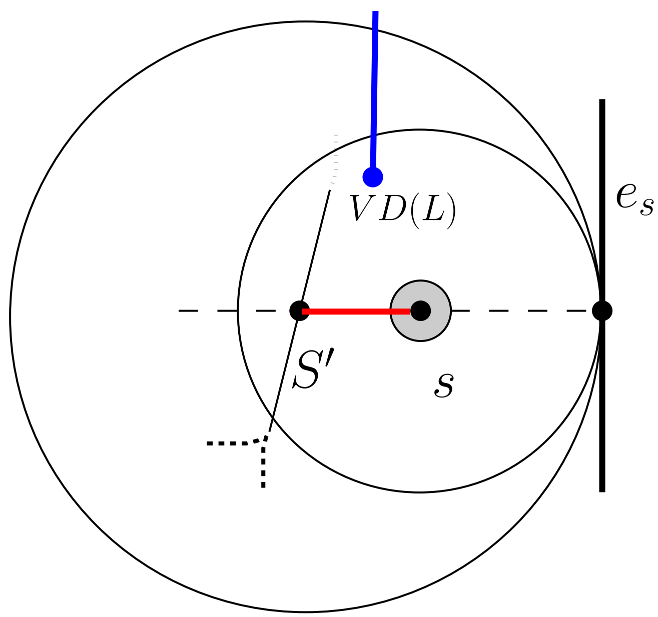
Kreisförmiger! Voronoi Diagramme 2.1.1

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren: Möglichst großer Abstand



Start s' kann stets angelaufen werden

- s in Region von e_s, Kreis frei
- Kürzester Weg zu e_s, Strahl Richtung Bisector
- Trifft Bisector bei s', Weg ist frei!!



Algorithmus 2.1: Kreisförmiger Roboter

Vorbereitung:

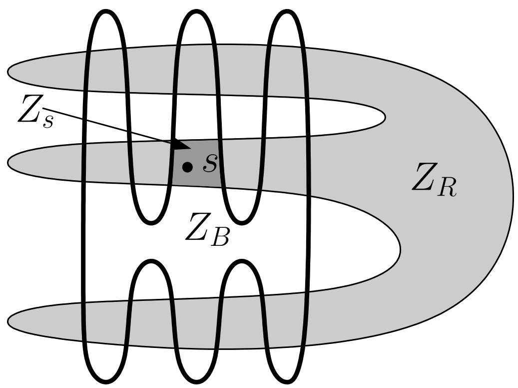
- Konstruiere das Voronoi–Diagramm $VD(L)$, Kanten der Hindernisse als Liniensegmente: $O(n \log n)$
- Point–Location Trapezzerlegung: $O(n \log n)$

Query für zwei Punkte s, t:

- Lokalisiere s, t im VD: $O(\log n)$
- s gehört zu e_s, t gehört zu e_t
- Bestimme mit Strahlen Punkt t' und s' auf $VD(L)$: $O(1)$
- Berechne (BFS) in $VD(L)$ Pfad von s' nach t' mit zul. Mindestabstand oder berichte, dass es keinen gibt: $O(n)$

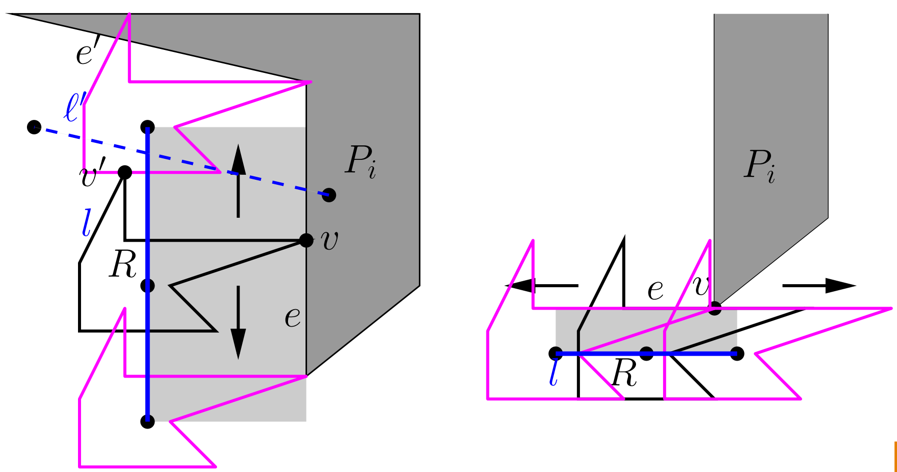
Kreisförmig, konvex! Beliebiger Roboter!

- Bisher: Divide and Conquer Konfigurationsraum
- Nicht konvex: Komplexität $\Theta((nm)^2)$
- Merge komplett: $\Theta((nm)^2)$
- Beobachtung: Zelle Z_s nicht so komplex??



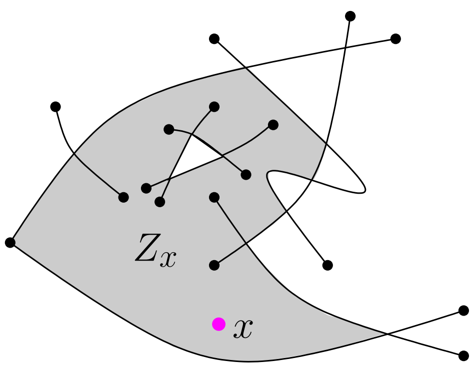
Begrenzung einer Zelle!!

- Nicht-konvexer Roboter R mit $|R| = m$. Polygonale Szene n Ecken
- Ecke Roboter, Kante Hindernis
- Ecke Hindernis, Kante Roboter
- $\Theta(mn)$ viele Kanten



Allgemeiner!!

- Menge von Segmenten
- Je zwei schneiden sich s mal
- Spezieller Punkt x
- Komplexität der Zelle Z_x



Fahrplan!!

- Divide and Conquer!!
- Teile Segmente in zwei gleichgroße Mengen Z_1, Z_2
- Berechne Z_{1_x} und Z_{2_x}
- Merge zu $\{Z_1 \cup Z_2\}_x$
- Spezieller Merge wegen Schnitt mit x
- RED BLUE Merge
- Merge: Komplexität des Ergebnisses

Exkurs: Davenport-Schinzel-Sequenzen

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich
- keine zwei verschiedenen Buchstaben wechseln mehr als s mal

Bsp.: ABRAKADABRA; max. 4 Wechsel (A und B)

$\lambda_s(n)$ maximale Länge eines solchen Wortes

Davenport-Schinzel-Sequenzen: Th. A9

Fast linear!! (Ohne Beweis!)

$$\begin{aligned}
\lambda_1(n) &= n \\
\lambda_2(n) &= 2n - 1 \\
\lambda_3(n) &\in \Theta(n\,\alpha(n)) \\
\lambda_4(n) &\in \Theta(n \cdot 2^{\alpha(n)}) \\
\lambda_s(n) &\in O(n \log^*(n)) \in O(n^2)
\end{aligned}$$

$\alpha(n)$ Inverse Ackermann Fkt.

$\log^*(n)$

Untere Kontur: Def. A10

$f_1, f_2, \ldots, f_n$ reellwertige Funktionen, entweder

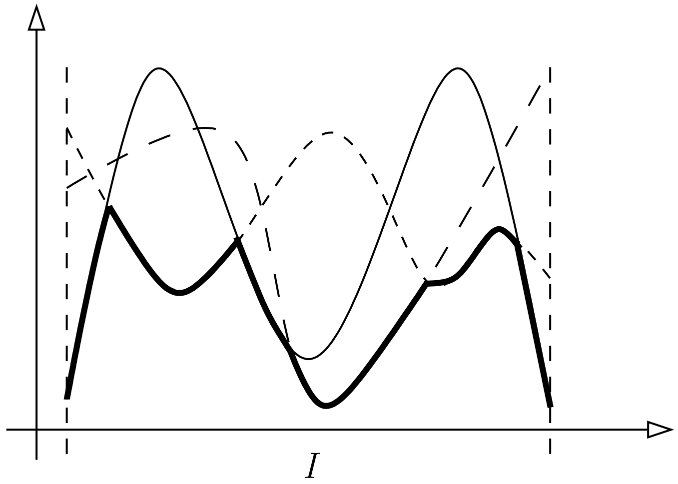
- über einem gemeinsamen Intervall I oder
- über je einem Intervall $I_j \subseteq I$, $1 \leq j \leq n$.

Je zwei Funktionen max. s gemeinsame Schnittpunkte

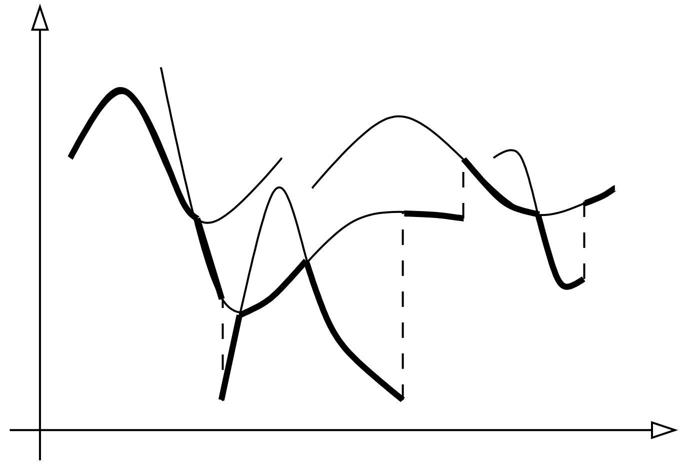
$L(x) := \min \{ f_i(x) | 1 \leq i \leq n \}$

Lower envelope der Funktionsgraphen

Untere Kontur: Th. A12



(1)


(2)

$L(x)$ besteht aus maximal

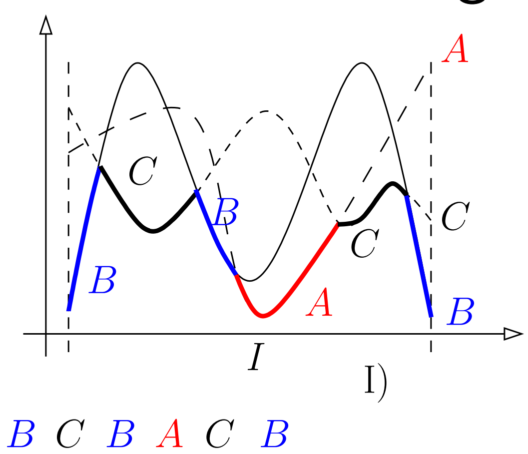
1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

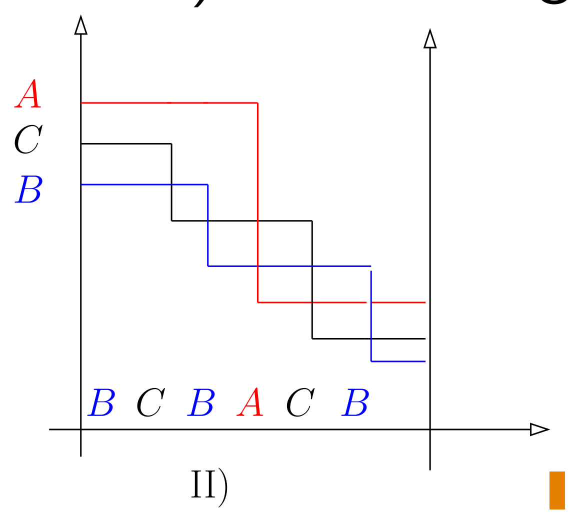
vielen Teilstücken

Beweis: Th. A12

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

1.: I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



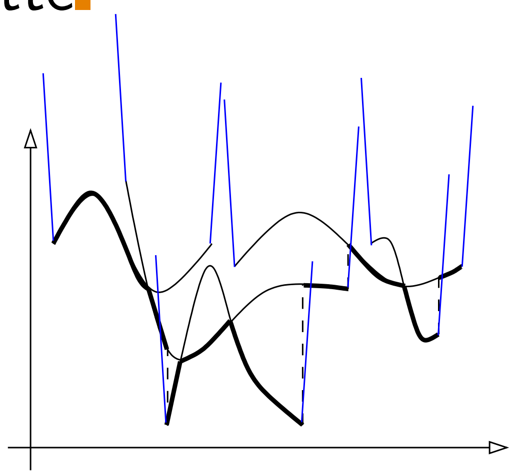


Beweis: Th. A12

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden

Max zwei zusätzliche Schnitte



(2)

Zellen: Th. 2.18

A Arrangement von n Kurvenstücken von denen sich zwei nur s mal schneiden. Jede Zelle von A hat Komplexität $O(\lambda_{s+2}(n))$.

Weniger als $\Omega(n^2)$!! Beweis!!