

Vorlesung Algorithmische Geometrie

Anwendungen der WSPD & Sichtbarkeitsgraphen

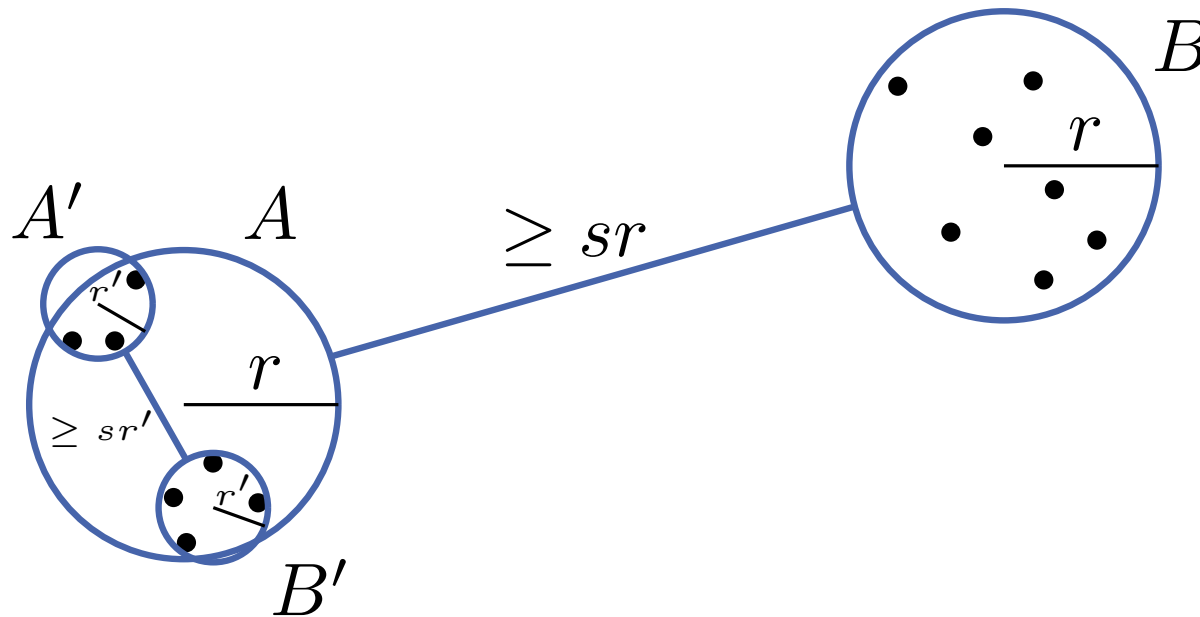
INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg
10.07.2012



Wdh.: Well-Separated Pair Decomposition

Def.: Ein Paar disjunkter Punktmenge A und B im \mathbb{R}^d heißt **s -well separated** für ein $s > 0$, falls A und B jeweils von einer Kugel mit Radius r überdeckt werden und der Abstand der beiden Kugeln mindestens sr ist.



Wdh.: Well-Separated Pair Decomposition

Def.: Ein Paar disjunkter Punktmenge A und B im \mathbb{R}^d heißt **s -well separated** für ein $s > 0$, falls A und B jeweils von einer Kugel mit Radius r überdeckt werden und der Abstand der beiden Kugeln mindestens sr ist.

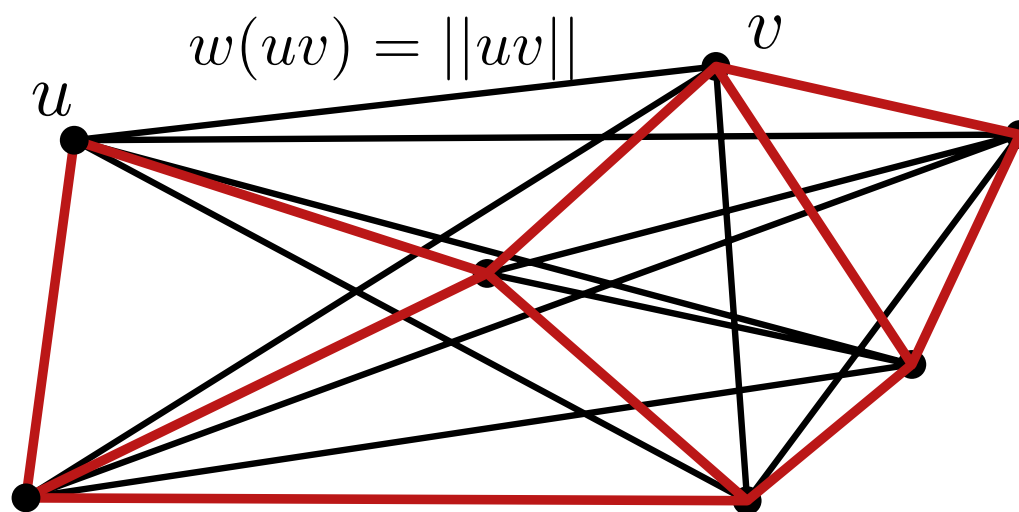
Def.: Für eine Punktmenge P und ein $s > 0$ ist eine **s -well separated pair decomposition** (s -WSPD) eine Menge von Paaren $\{\{A_1, B_1\}, \dots, \{A_m, B_m\}\}$ mit

- $A_i, B_i \subset P$ für alle i
- $A_i \cap B_i = \emptyset$ für alle i
- $\bigcup_{i=1}^m A_i \otimes B_i = P \otimes P$
- $\{A_i, B_i\}$ s -well separated für alle i

Wdh.: t -Spanner

Für eine Menge P von n Punkten in \mathbb{R}^d ist der **Euklidische Graph** $\mathcal{EG}(P) = (P, \binom{P}{2})$ der vollständige gewichtete Graph, dessen Kantengewichte dem Euklidischen Abstand der Kantenendpunkte entsprechen.

Da $\mathcal{EG}(P) \Theta(n^2)$ Kanten hat, wird oft ein dünner Graph mit $O(n)$ Kanten gesucht, dessen kürzeste Wege die Kantengewichte in $\mathcal{EG}(P)$ approximieren.



Für eine Menge P von n Punkten in \mathbb{R}^d ist der **Euklidische Graph** $\mathcal{EG}(P) = (P, \binom{P}{2})$ der vollständige gewichtete Graph, dessen Kantengewichte dem Euklidischen Abstand der Kantenendpunkte entsprechen.

Da $\mathcal{EG}(P) \Theta(n^2)$ Kanten hat, wird oft ein dünner Graph mit $O(n)$ Kanten gesucht, dessen kürzeste Wege die Kantengewichte in $\mathcal{EG}(P)$ approximieren.

Def.: Ein gewichteter Graph G mit Knotenmenge P heißt **t -Spanner** für P und einen Dehnungsfaktor $t \geq 1$, falls für alle Paare $x, y \in P$ gilt

$$\|xy\| \leq \delta_G(x, y) \leq t \cdot \|xy\|,$$

wobei $\delta_G(x, y) =$ Länge kürzester x - y -Weg in G .

Wdh.: WSPD und t -Spanner

Def.: Für n Punkte P in \mathbb{R}^d und eine WSPD W von P definiere den Graphen $G = (P, E)$ mit
 $E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \text{rep}(u) = x, \text{rep}(v) = y\}$.

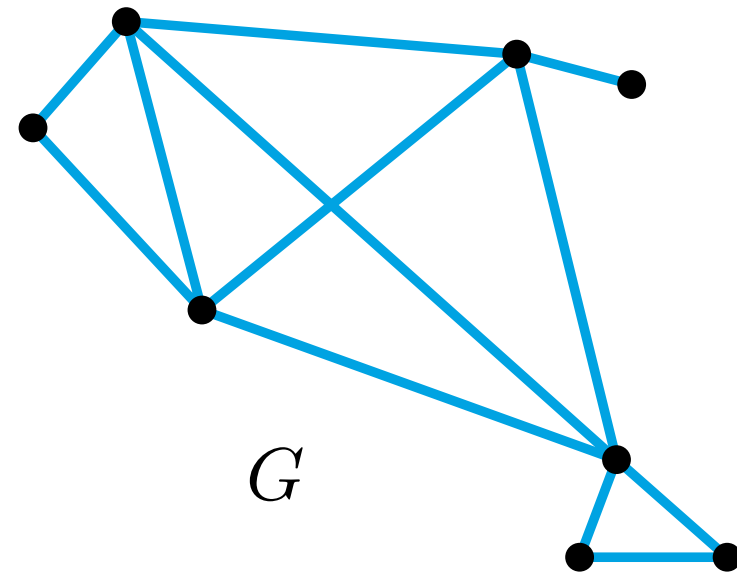
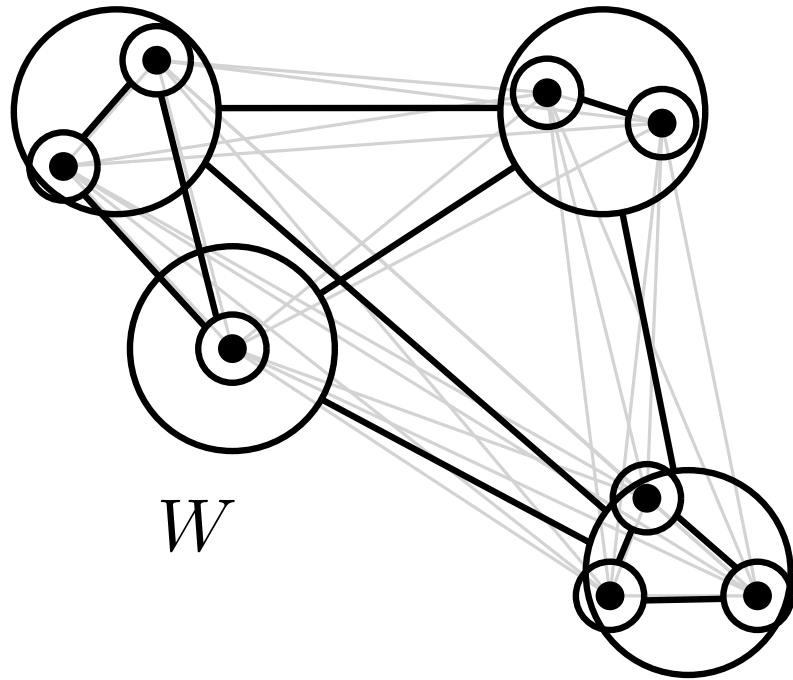
Wdh.: Jedes Paar $\{u, v\} \in W$ entspricht zwei Quadtreeknoten u und v . Aus jedem Quadtreeknoten wird wie folgt ein Repräsentant gewählt. Für Blatt u definiere den Repräsentanten

$$\text{rep}(u) = \begin{cases} p & \text{falls } P_u = \{p\} \text{ (} u \text{ ist Blatt)} \\ \emptyset & \text{sonst.} \end{cases}$$

Für einen inneren Knoten v setze $\text{rep}(v) = \text{rep}(u)$ für ein nichtleeres Kind u von v .

Wdh.: WSPD und t -Spanner

Def.: Für n Punkte P in \mathbb{R}^d und eine WSPD W von P definiere den Graphen $G = (P, E)$ mit
 $E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \text{rep}(u) = x, \text{rep}(v) = y\}$.



Wdh.: WSPD und t -Spanner

Def.: Für n Punkte P in \mathbb{R}^d und eine WSPD W von P definiere den Graphen $G = (P, E)$ mit
$$E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \text{rep}(u) = x, \text{rep}(v) = y\}.$$

Lemma: Ist W eine s -WSPD für ein geeignetes $s = s(t)$, so ist G ein t -Spanner für P mit $O(s^d n)$ Kanten.

Satz: Für eine Menge P von n Punkten in \mathbb{R}^d und ein $\varepsilon \in (0, 1]$ kann ein $(1 + \varepsilon)$ -Spanner für P mit $O(n/\varepsilon^d)$ Kanten in $O(n \log n + n/\varepsilon^d)$ Zeit berechnet werden.

Problem: Finde für eine Punktmenge P einen minimalen Spannbaum (MST) im Euklidischen Graphen $\mathcal{EG}(P)$.

Prim: MST in einem Graph $G = (V, E)$ kann in $O(|E| + |V| \log |V|)$ Zeit berechnet werden.

- $\mathcal{EG}(P)$ hat $\Theta(n^2)$ Kanten \Rightarrow Laufzeit $\Theta(n^2)$:-)
- $(1 + \varepsilon)$ -Spanner für P hat $O(n/\varepsilon^d)$ Kanten \Rightarrow Laufzeit $O(n \log n + n/\varepsilon^d)$:-)

Wie gut ist der MST eines $(1 + \varepsilon)$ -Spanners?

Satz 1: Der aus einem $(1 + \varepsilon)$ -Spanner von P gewonnene MST ist eine $(1 + \varepsilon)$ -Approximation des EMST von P .

Durchmesser von P

Problem: Finde den Durchmesser einer Punktmenge P , d.h. das Paar $\{x, y\} \subset P$ mit größtem Abstand.

- brute-force alle Punktpaare testen \Rightarrow Laufzeit $O(n^2)$:-)
- teste Abstände $\| \text{rep}(u) - \text{rep}(v) \|$ aller ws-Paare $\{P_u, P_v\}$
 \Rightarrow Laufzeit $O(n \log n + s^d n)$:-)

Wie gut ist der berechnete Durchmesser?

Satz 2: Der aus einer s -WSPD von P für $s = 4/\varepsilon$ berechnete Durchmesser ist eine $(1 + \varepsilon)$ -Approximation des Durchmessers von P .

Welche weiteren Anwendungen hat die WSPD?

WSPD bietet sich immer dann an, wenn man auf die $\Theta(n^2)$ exakten Distanzen in einer Punktmenge verzichten kann und diese stattdessen approximiert. Ein Beispiel sind kräftebasierte Layoutalgorithmen im Graphenzeichnen, wo paarweise Abstoßungskräfte von n Objekten berechnet werden müssen.

Wozu geometrisch approximieren?

Einerseits ersetzt man dadurch langsame Berechnungen durch schnellere (aber ungenauere), andererseits sind oft auch die Eingabedaten schon mit einer gewissen Ungenauigkeit behaftet, so dass approximative Lösungen je nach Anwendung ausreichend sind.

Geht es nicht auch genauso schnell exakt?

Oft im \mathbb{R}^2 ja, aber nicht mehr im \mathbb{R}^d für $d > 2$. (EMST, Durchmesser)

Rückblick und Prüfungsvorbereitung

Was haben wir bisher alles behandelt?

t -Spanner

Ebenenunterteilung

konvexe Hülle (2d und 3d)

Quadtrees

randomisierte Algorithmen

kd-Trees

Voronoi-Diagramme

Dualität

Polygontriangulierung

Streckenschnitte

WSPD

Schnitt von Halbebenen

Delaunay-Triangulierung

Lineares Programmieren

Sweep-Line Algorithmen

Trapezzerlegung

Range-Trees

Geradenarrangements

Divide & Conquer

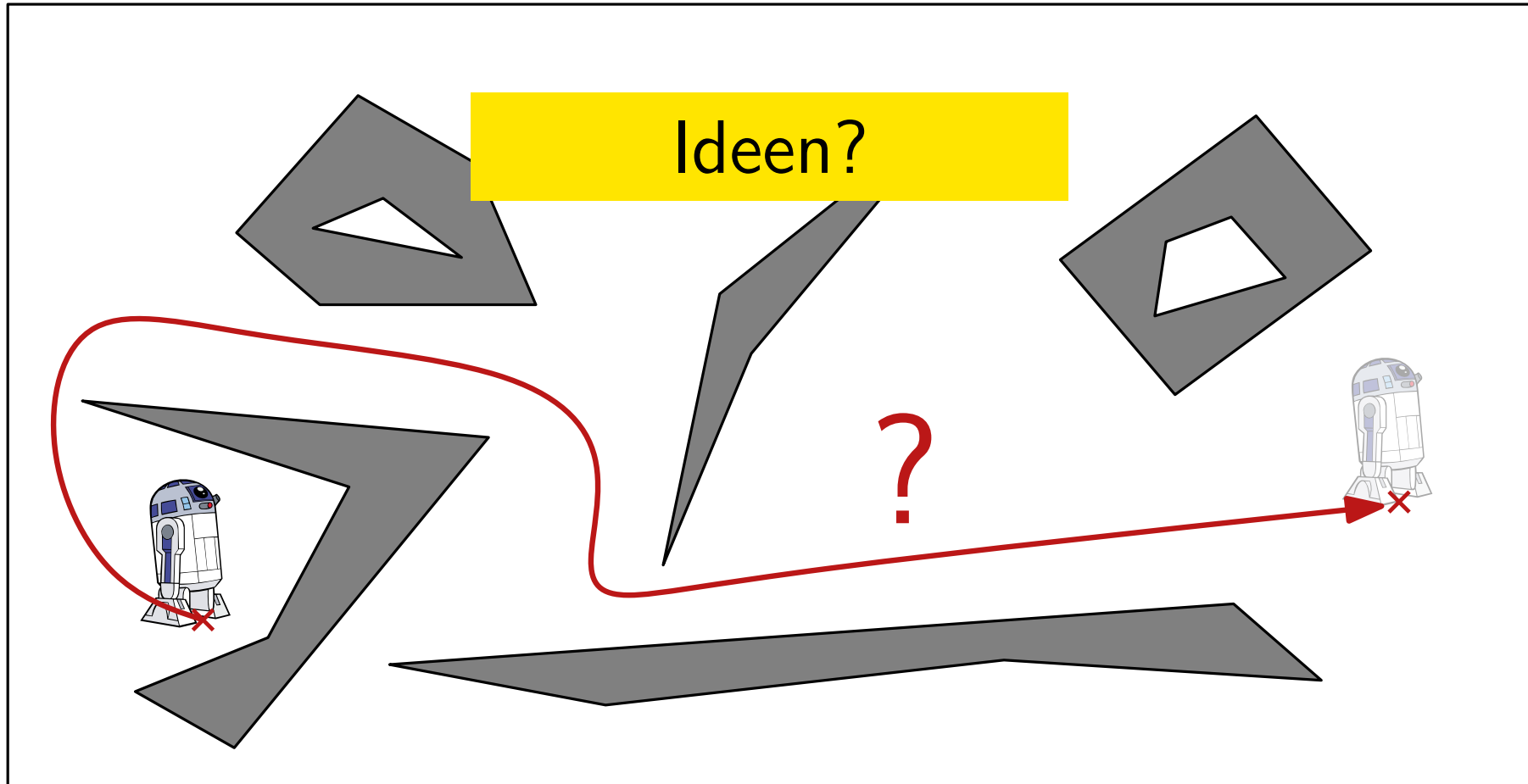
Was kommt in der Prüfung dran?

- grundlegende Problemdefinitionen
- Aufbau und Queries von Datenstrukturen
- Zusammenhänge zwischen verschiedenen Datenstrukturen
- Eigenschaften und Ablauf der Algorithmen
- zentrale Beweisideen, nicht die genauen technischen Details
- Einsatz der Algorithmen und Datenstrukturen in Anwendungen
- bei 3SWS bzw. im Master: Stoff der Übungen

Wie vorbereiten?

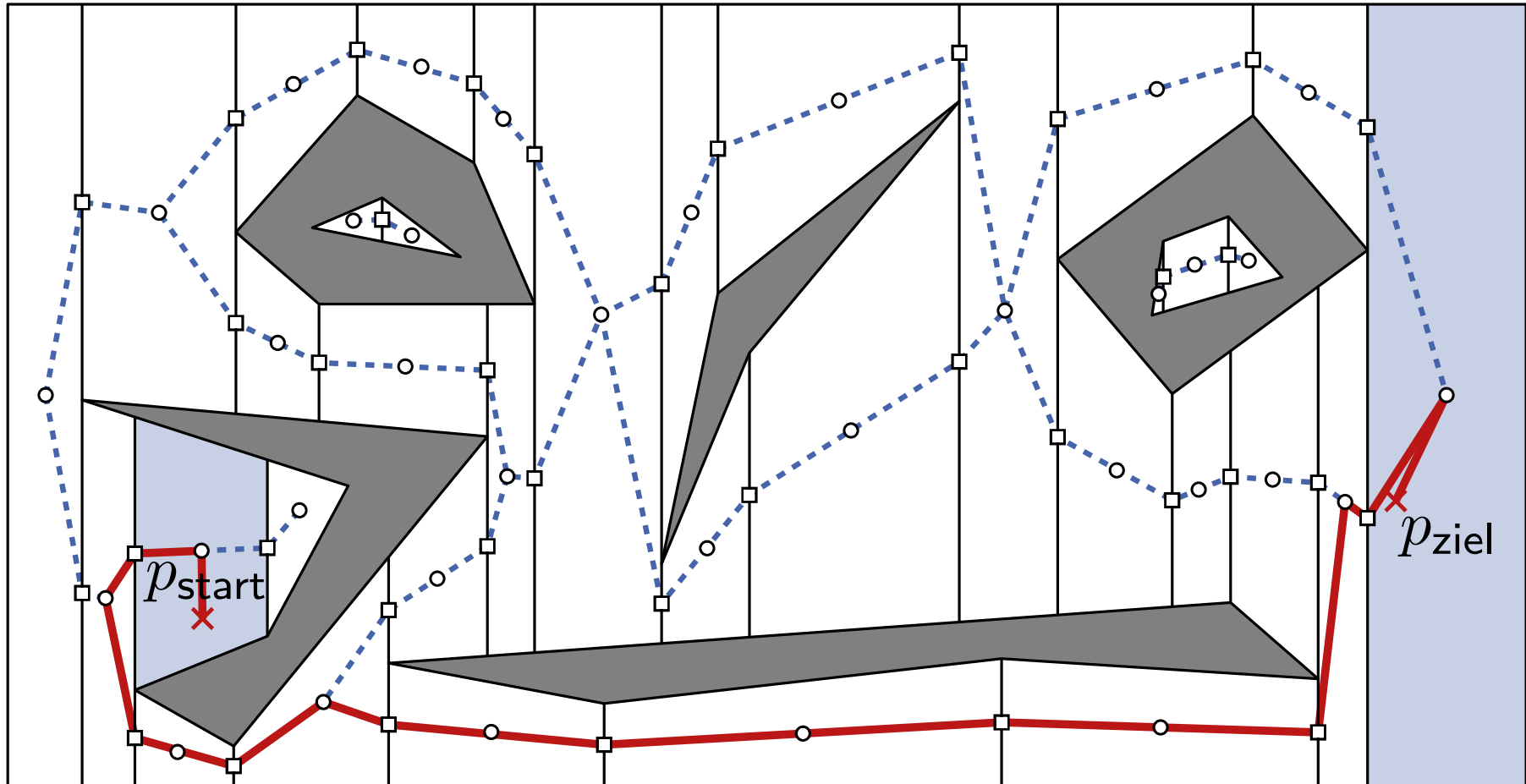
- Vorlesungsfolien **und** Tafelbeweise aus angegebener Literatur (v.a. [BCKO08] und tlws. Skript [M10])
- Übungsaufgaben
- *ergänzend* nicht behandeltes Material aus der Literatur

Sichtbarkeitsgraphen



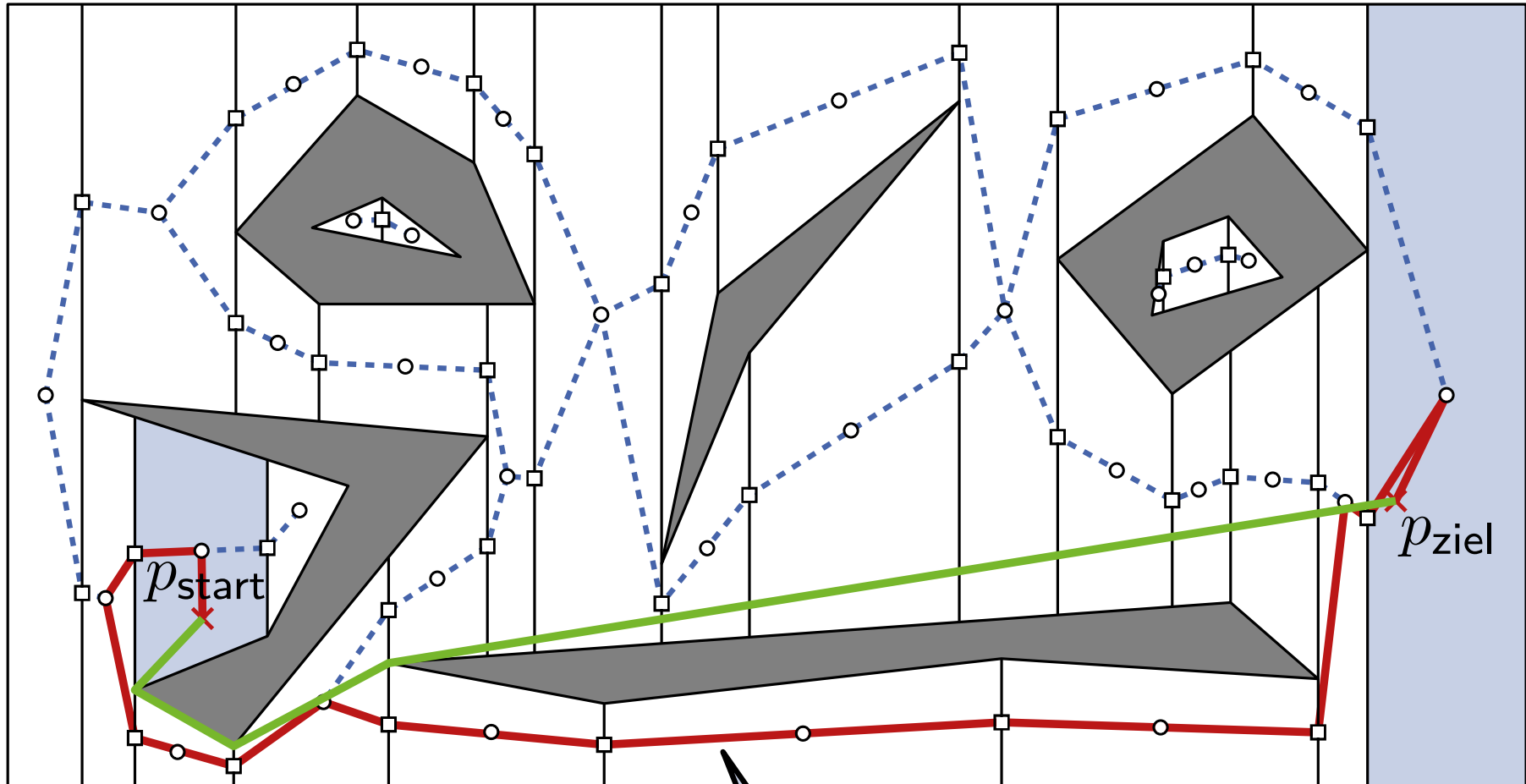
Problem: Gegeben ein (punktförmiger) Roboter an Position p_{start} in einem Gebiet mit polygonalen Hindernissen finde einen möglichst kurzen Weg zum Ziel p_{ziel} um die Hindernisse herum.

Erste Idee: Kürzeste Wege in Graphen



- erstelle Trapezzerlegung
- entferne Segmente in Hindernissen
- Knoten in Trapezen und Vertikalen
- euklidisch gewichteter „Dualgraph“ G mit Viaknoten auf Vertikalen
- Lokalisierere Start und Ziel
- kürzester Weg mit Dijkstra in G

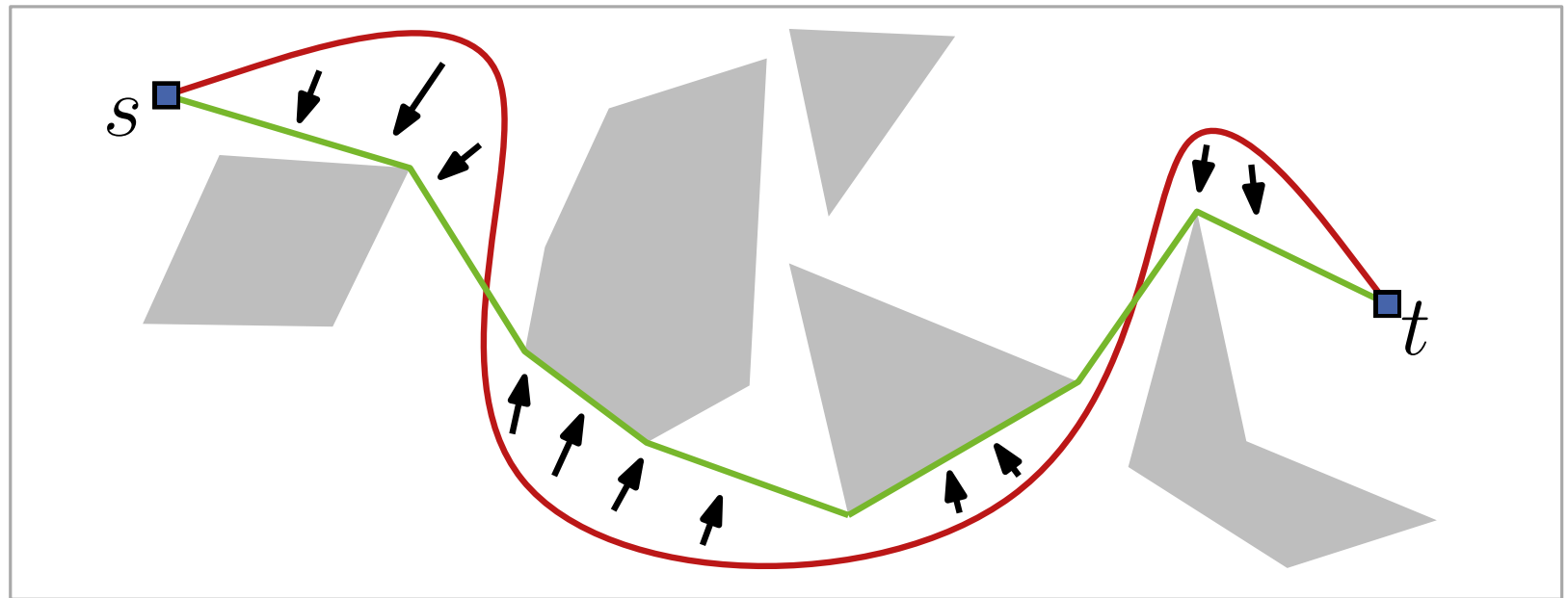
Erste Idee: Kürzeste Wege in Graphen



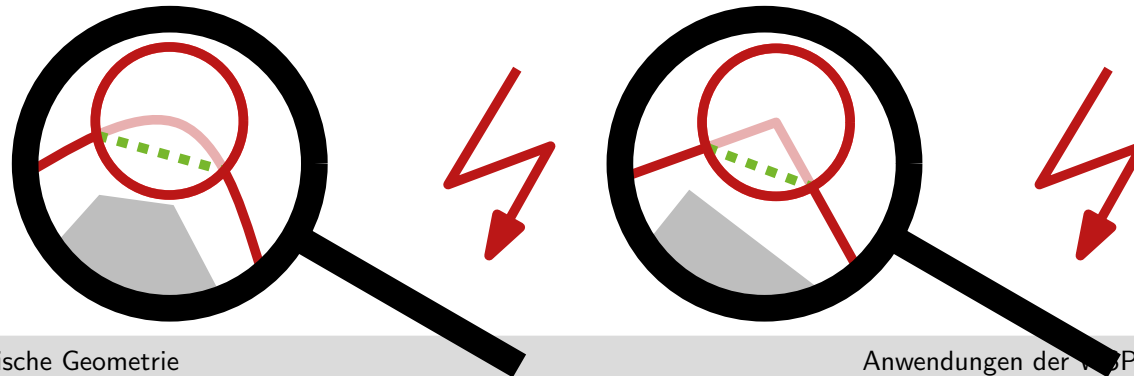
- erstelle Trapezzerlegung
 - entferne Segmente in Hindernissen
 - Knoten in Trapezen und Vertikalen
 - euklidisch gewichteter „Dualgraph“ G mit Viaknoten auf Vertikalen
- kein kürzester Weg!
- Lokalisierere Start und Ziel
kürzester Weg mit Dijkstra in G

Kürzeste Wege in Polygonegebieten

Lemma 1: Für eine Menge S von disjunkten Polygonen in \mathbb{R}^2 und zwei Punkte s und t außerhalb S ist jeder kürzeste st -Weg in $\mathbb{R}^2 \setminus \bigcup S$ ein Polygonzug dessen innere Knoten Knoten von S sind.



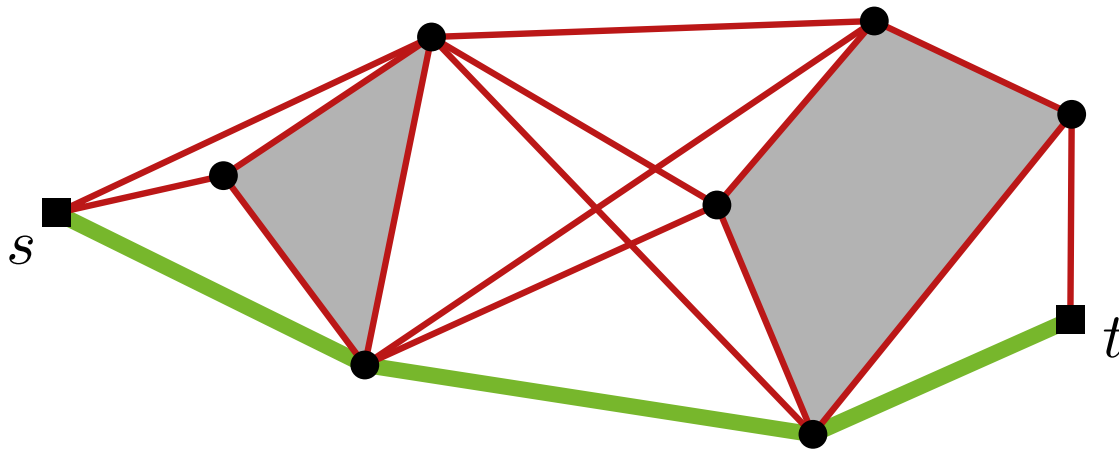
Beweisskizze:



Sichtbarkeitsgraph

Gegeben sei eine Menge S disjunkter offener Polygone...

...mit Knotenmenge $V(S)$.



Def.: Dann ist $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ der **Sichtbarkeitsgraph** von S mit $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S) \text{ und } u \text{ sieht } v\}$ und $w(uv) = |uv|$.
Dabei gilt u **sieht** $v : \Leftrightarrow \overline{uv} \cap \bigcup S = \emptyset$

Definiere $S^* = S \cup \{s, t\}$ und $G_{\text{vis}}(S^*)$ analog.

Lemma 1

\Rightarrow

Der kürzeste st -Weg, der die Hindernisse in S vermeidet, entspricht einem kürzesten Weg in $G_{\text{vis}}(S^*)$.

SHORTESTPATH(S, s, t)

$$n = |V(S)|, m = |E_{\text{vis}}(S)|$$

Input: Hindernismenge S , Punkte $s, t \in \mathbb{R}^2 \setminus \bigcup S$

Output: kürzester kollisionsfreier st -Weg in S

- 1 $G_{\text{vis}} \leftarrow \text{VISIBILITYGRAPH}(S \cup \{s, t\})$ $O(n^2 \log n)$
 - 2 **foreach** $uv \in E_{\text{vis}}$ **do** $w(uv) \leftarrow |uv|$ $O(m)$
 - 3 **return** $\text{DIJKSTRA}(G_{\text{vis}}, w, s, t)$ $O(n \log n + m)$
-
- $O(n^2 \log n)$

Satz 1: Ein kürzester st -Weg in einem Gebiet mit Polygon-Hindernissen mit n Kanten kann in $O(n^2 \log n)$ Zeit berechnet werden.

Sichtbarkeitsgraph berechnen

VISIBILITYGRAPH(S)

Input: Menge disjunkter Polygone S

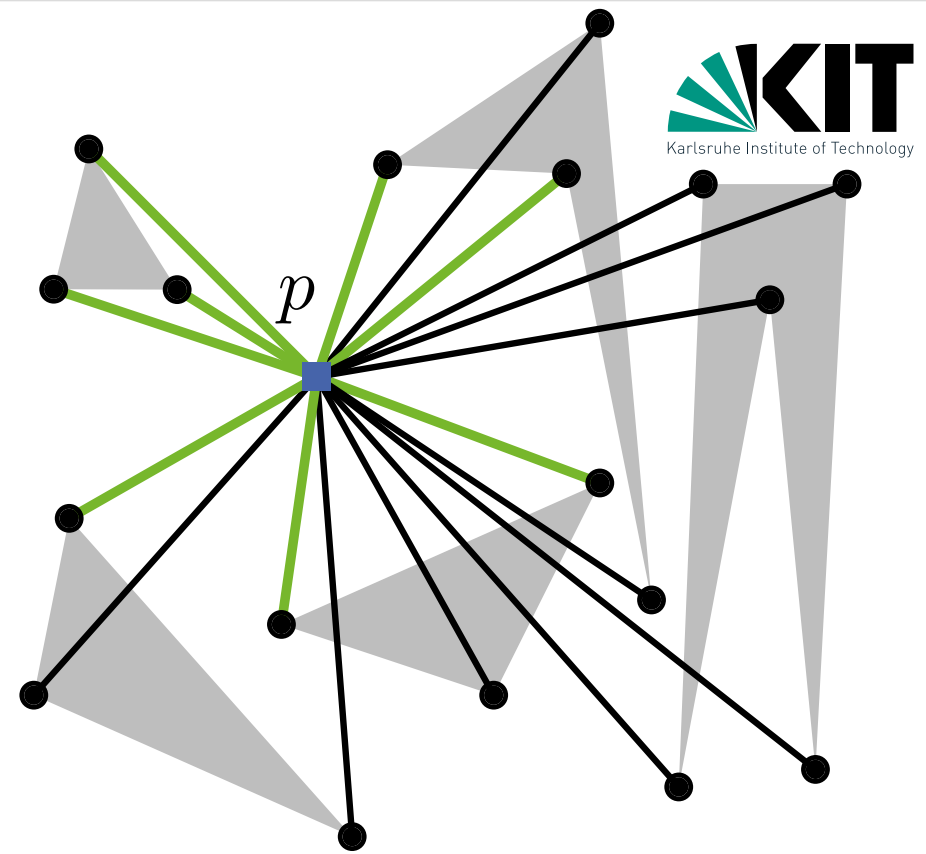
Output: Sichtbarkeitsgraph $G_{\text{vis}}(S)$

```
1  $E \leftarrow \emptyset$ 
2 foreach  $v \in V(S)$  do
3    $W \leftarrow \text{VISIBLEVERTICES}(v, S)$ 
4    $E \leftarrow E \cup \{vw \mid w \in W\}$ 
5 return  $E$ 
```

Sichtbare Knoten berechnen

$\text{VISIBLEVERTICES}(p, S)$

Aufgabe: Gegeben p und S
finde in $O(n \log n)$ Zeit alle
von p aus sichtbaren Knoten
in $V(S)$!



Sichtbare Knoten berechnen

$\text{VISIBLEVERTICES}(p, S)$

$$r \leftarrow \{p + k \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mid k \in \mathbb{R}_0^+\}$$

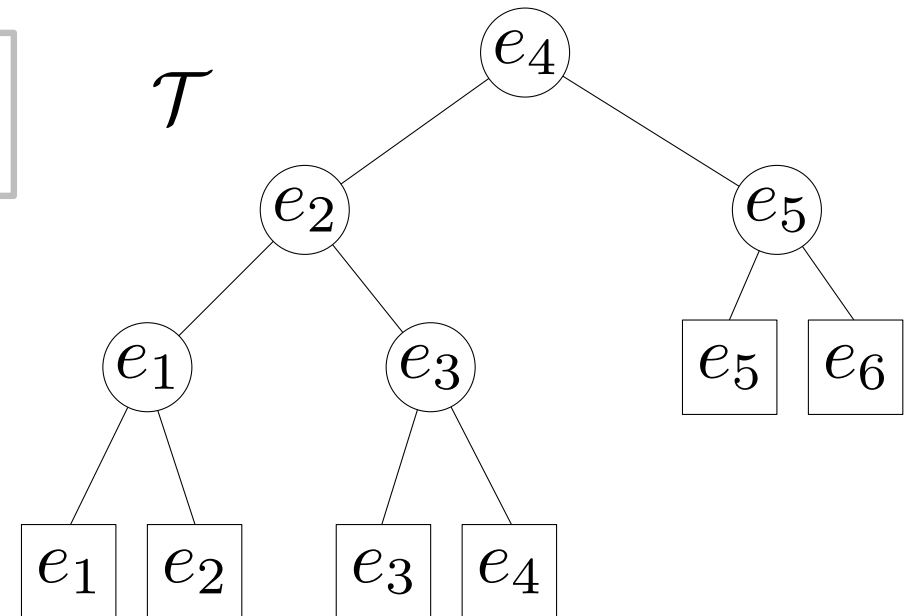
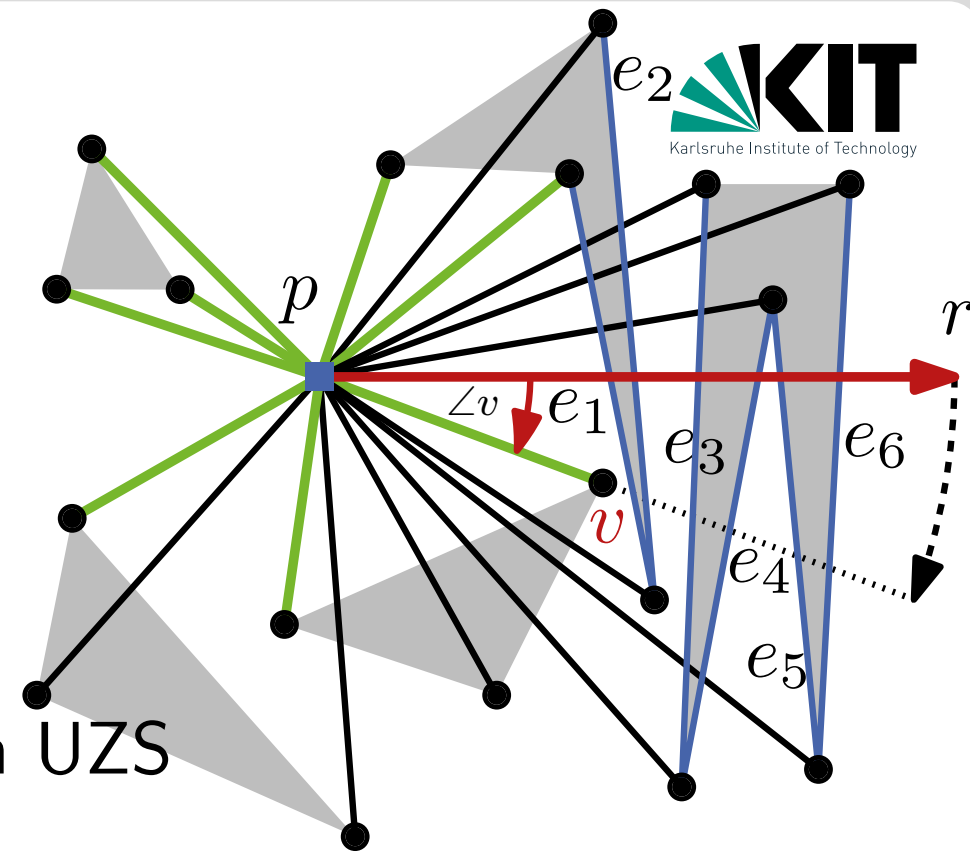
$$I \leftarrow \{e \in E(S) \mid e \cap r \neq \emptyset\}$$

$$\mathcal{T} \leftarrow \text{balancedBinaryTree}(I)$$

$$w_1, \dots, w_n \leftarrow \text{sortiere } V(S) \text{ im UZS}$$

$$v \prec v' :\Leftrightarrow$$

$$\begin{aligned} &\angle v < \angle v' \text{ or} \\ &(\angle v = \angle v' \text{ and } |pv| < |pv'|) \end{aligned}$$



Sweep-Verfahren mit Rotation

Sichtbare Knoten berechnen

$\text{VISIBLEVERTICES}(p, S)$

$$r \leftarrow \{p + k \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mid k \in \mathbb{R}_0^+\}$$

$$I \leftarrow \{e \in E(S) \mid e \cap r \neq \emptyset\}$$

$$\mathcal{T} \leftarrow \text{balancedBinaryTree}(I)$$

$$w_1, \dots, w_n \leftarrow \text{sortiere } V(S) \text{ im UZS}$$

$$W \leftarrow \emptyset$$

for $i = 1$ **to** n **do**

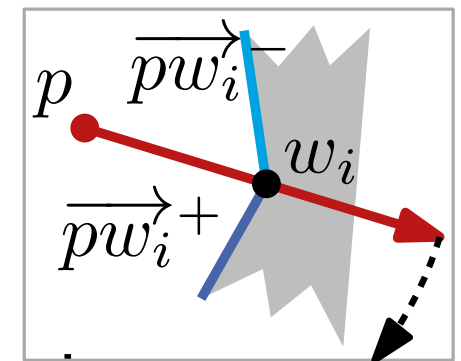
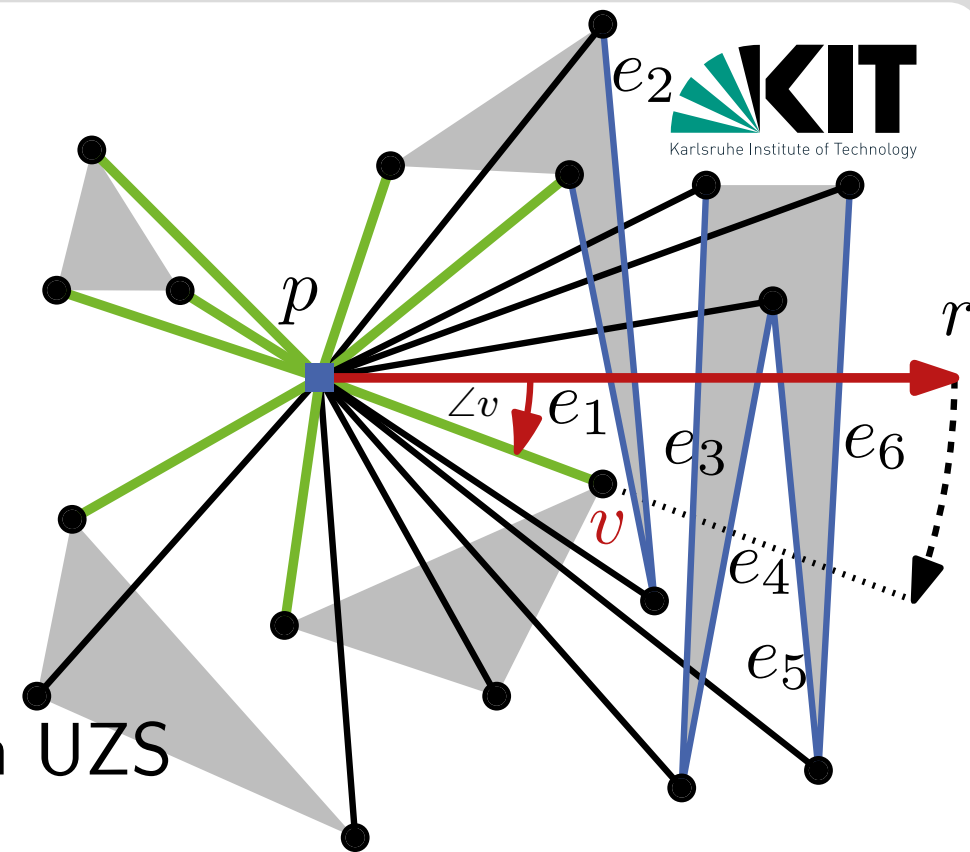
if $\text{VISIBLE}(p, w_i)$ **then**

$$\quad W \leftarrow W \cup \{w_i\}$$

füge in \mathcal{T} zu w_i inzidente Kante aus $\overrightarrow{pw_i}^+$ ein

lösche aus \mathcal{T} zu w_i inzidente Kanten aus $\overrightarrow{pw_i}^-$

return W



Fallunterscheidung Sichtbarkeit

$VISIBLE(p, w_i)$

if $\overline{pw_i}$ schneidet Polygon von w_i **then**
 | **return false**

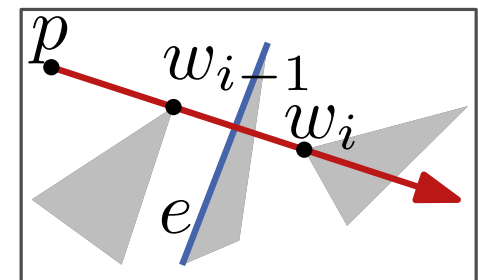
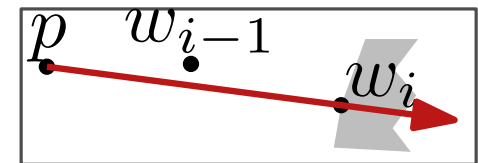
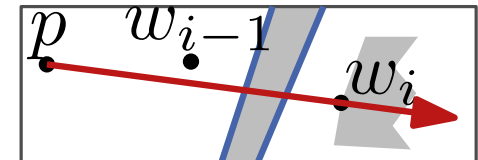
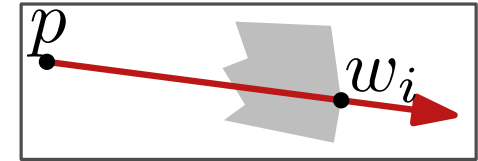
if $i = 1$ oder $w_{i-1} \notin \overline{pw_i}$ **then**
 | $e \leftarrow$ Kante im linkesten Blatt von \mathcal{T}
 | **if** $e \neq \text{nil}$ und $\overline{pw_i} \cap e \neq \emptyset$ **then**
 | **return false**
 | **else return true**

else

 | **if** w_{i-1} nicht sichtbar **then**
 | **return false**

 | **else**

 | $e \leftarrow$ suche Kante in \mathcal{T} , die $\overline{w_{i-1}w_i}$ schneidet
 | **if** $e \neq \text{nil}$ **then return false**
 | **else return true**



Satz 1: Ein kürzester st -Weg in einem Gebiet mit Polygon-Hindernissen mit n Kanten kann in $O(n^2 \log n)$ Zeit berechnet werden.

Beweis:

- Korrektheit folgt direkt aus Lemma 1
- Laufzeit:
 - `VISIBLEVERTICES` benötigt $O(n \log n)$ Zeit pro Knoten
 - n Aufrufe von `VISIBLEVERTICES` □

$O(n^2)$ mit Dualität!
(s. Übung oder Skript D. Mount [M10] Lect. 30)

Roboter sind meistens nicht punktförmig...

Für den Fall von Robotern, deren Grundfläche ein konvexes Polygon ist und die nicht rotieren können, geht es trotzdem durch geeignete Vergrößerung der Hindernisse (→ Minkowski-Summe, Kap. 13 in [BCKO08]).

Geht es schneller als $O(n^2 \log n)$?

Ja, durch Ausnutzung der Dualität und einen simultanen Rotations-Sweep für alle Punkte im dualen Geradenarrangement geht es auch in $O(n^2)$. Da G_{vis} $\Omega(n^2)$ Kanten haben kann, lässt sich der

Sichtbarkeitsgraph im Allgemeinen auch nicht schneller konstruieren.

Es gibt jedoch einen ausgabesensitiven $O(n \log n + m)$ -Algorithmus.

[Ghosh, Mount 1987]

Sucht man jedoch nur einen kürzesten Euklidischen st -Weg, gibt es einen Algorithmus mit optimaler Laufzeit $O(n \log n)$. [Hershberger, Suri 1999]

