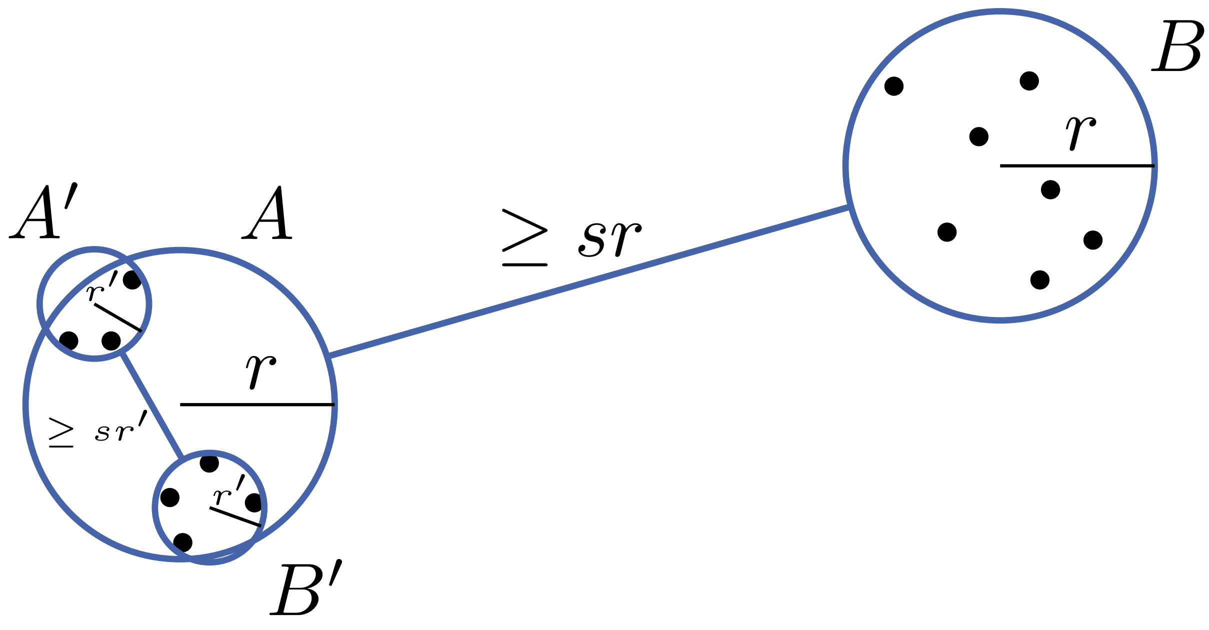
Computational Geometry Lecture
Applications of WSPD & Visibility Graphs

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze
25.05.2018

Recall: Well-Separated Pair Decomposition

Def: A pair of disjoint point sets A and B in $\mathbb{R}^d$ is called s-**well separated** for some $s > 0$, if A and B can each be covered by a ball of radius r whose distance is at least sr.



Recall: Well-Separated Pair Decomposition

Def: A pair of disjoint point sets A and B in $\mathbb{R}^d$ is called s-**well separated** for some $s > 0$, if A and B can each be covered by a ball of radius r whose distance is at least sr.

Def: For a point set P and some $s > 0$ an s-**well separated pair decomposition** (s-WSPD) is a set of pairs $\{\{A_1, B_1\}, \dots, \{A_m, B_m\}\}$ with

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

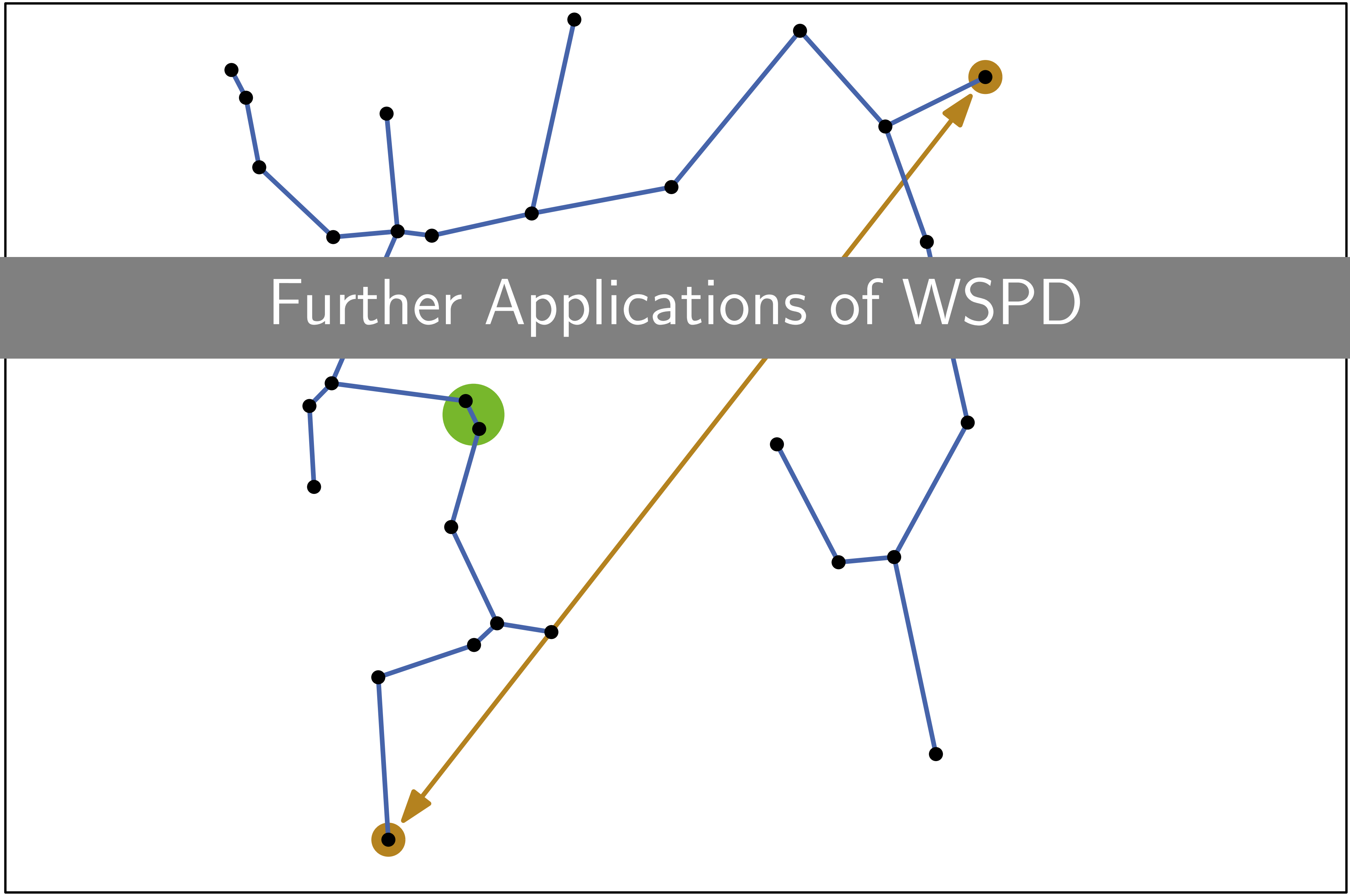
Recall: Well-Separated Pair Decomposition

Def: A pair of disjoint point sets A and B in $\mathbb{R}^d$ is called s-**well separated** for some $s > 0$, if A and B can each be covered by a ball of radius r whose distance is at least sr.

Def: For a point set P and some $s > 0$ an s-**well separated pair decomposition** (s-WSPD) is a set of pairs $\{\{A_1, B_1\}, \dots, \{A_m, B_m\}\}$ with

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

Thm 3: Given a point set P in $\mathbb{R}^d$ and $s \geq 1$ we can construct an s-WSPD with $O(s^d n)$ pairs in time $O(n \log n + s^d n)$.



Further Applications of WSPD

Euclidean MST

Problem: Given a point set P find a minimum spanning tree (MST) in the Euclidean graph $\mathcal{EG}(P)$.

Euclidean MST

Problem: Given a point set P find a minimum spanning tree (MST) in the Euclidean graph $\mathcal{EG}(P)$.

Prim: MST in a graph $G = (V, E)$ can be computed in $O(|E| + |V| \log |V|)$ time.

Euclidean MST

Problem: Given a point set P find a minimum spanning tree (MST) in the Euclidean graph $\mathcal{EG}(P)$.

Prim: MST in a graph $G = (V, E)$ can be computed in $O(|E| + |V| \log |V|)$ time.

- $\mathcal{EG}(P)$ has $\Theta(n^2)$ edges $\Rightarrow$ running time $O(n^2)$:-(
- $(1+\varepsilon)$-spanner for P has $O(n/\varepsilon^d)$ edges
  $\Rightarrow$ running time $O(n \log n + n/\varepsilon^d)$:-)

Euclidean MST

Problem: Given a point set P find a minimum spanning tree (MST) in the Euclidean graph $\mathcal{EG}(P)$.

Prim: MST in a graph $G = (V, E)$ can be computed in $O(|E| + |V| \log |V|)$ time.

- $\mathcal{EG}(P)$ has $\Theta(n^2)$ edges $\Rightarrow$ running time $O(n^2)$:-(
- $(1+\varepsilon)$-spanner for P has $O(n/\varepsilon^d)$ edges
  $\Rightarrow$ running time $O(n \log n + n/\varepsilon^d)$:-)

How good is the MST of a $(1+\varepsilon)$-spanner?

Euclidean MST

Problem: Given a point set P find a minimum spanning tree (MST) in the Euclidean graph $\mathcal{EG}(P)$.

Prim: MST in a graph $G = (V, E)$ can be computed in $O(|E| + |V| \log |V|)$ time.

- $\mathcal{EG}(P)$ has $\Theta(n^2)$ edges $\Rightarrow$ running time $O(n^2)$:-(
- $(1+\varepsilon)$-spanner for P has $O(n/\varepsilon^d)$ edges
  $\Rightarrow$ running time $O(n \log n + n/\varepsilon^d)$:-)

How good is the MST of a $(1+\varepsilon)$-spanner?

Thm 5: The MST obtained from a $(1+\varepsilon)$-spanner of P is a $(1+\varepsilon)$-approximation of the EMST of P.

Diameter of P

Problem: Find the diameter of a point set P (i.e., the pair $\{x, y\} \subset P$ with maximum distance).

Diameter of P

Problem: Find the diameter of a point set P (i.e., the pair $\{x, y\} \subset P$ with maximum distance).

- brute-force testing all point pairs $\Rightarrow$ running time $O(n^2)$:-(
- test distances $||\text{rep}(u)\ \text{rep}(v)||$ of all ws-pairs $\{P_u, P_v\}$
  $\Rightarrow$ running time $O(n \log n + s^d n)$:-)

Diameter of P

Problem: Find the diameter of a point set P (i.e., the pair $\{x, y\} \subset P$ with maximum distance).

- brute-force testing all point pairs $\Rightarrow$ running time $O(n^2)$:-(
- test distances $||\text{rep}(u)\ \text{rep}(v)||$ of all ws-pairs $\{P_u, P_v\}$
  $\Rightarrow$ running time $O(n \log n + s^d n)$:-)

How good is the computed diameter?

Diameter of P

Problem: Find the diameter of a point set P (i.e., the pair $\{x, y\} \subset P$ with maximum distance).

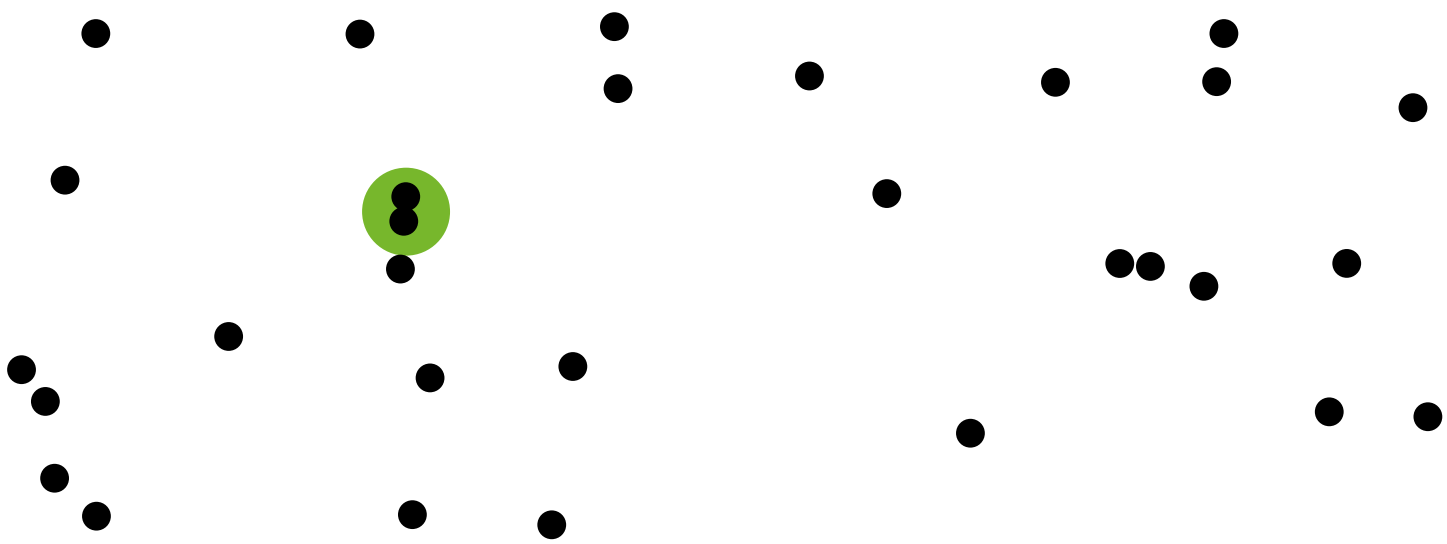
- brute-force testing all point pairs $\Rightarrow$ running time $O(n^2)$:-(
- test distances $||\text{rep}(u)\ \text{rep}(v)||$ of all ws-pairs $\{P_u, P_v\}$ $\Rightarrow$ running time $O(n \log n + s^d n)$:-)

How good is the computed diameter?

Thm 6: The diameter obtained from an s-WSPD of P for $s = 4/\varepsilon$ is a $(1 + \varepsilon)$-approximation of the diameter of P.

Closest Pair of Points

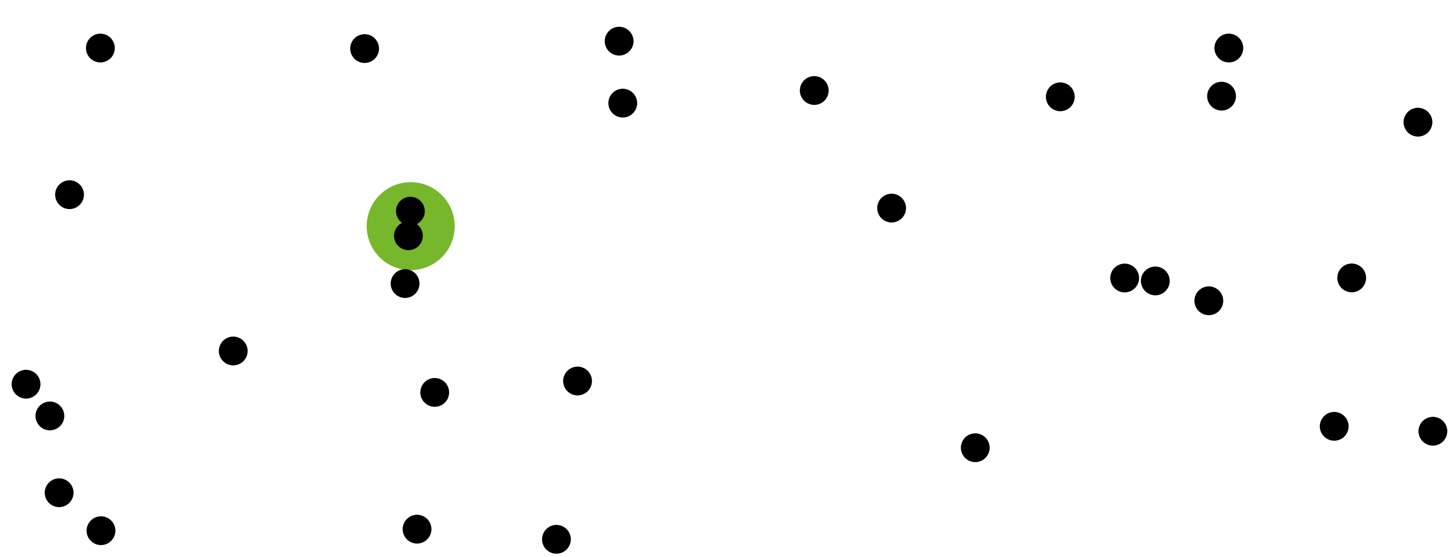
Problem: Find the pair $\{x, y\} \subset P$ with minimum distance.



Closest Pair of Points

Problem: Find the pair $\{x, y\} \subset P$ with minimum distance.

- brute-force testing all point pairs $\Rightarrow$ running time $O(n^2)$:-(
- test distances $||\text{rep}(u)\ \text{rep}(v)||$ of all ws-pairs $\{P_u, P_v\}$
  $\Rightarrow$ running time $O(n \log n + s^d n)$:-)

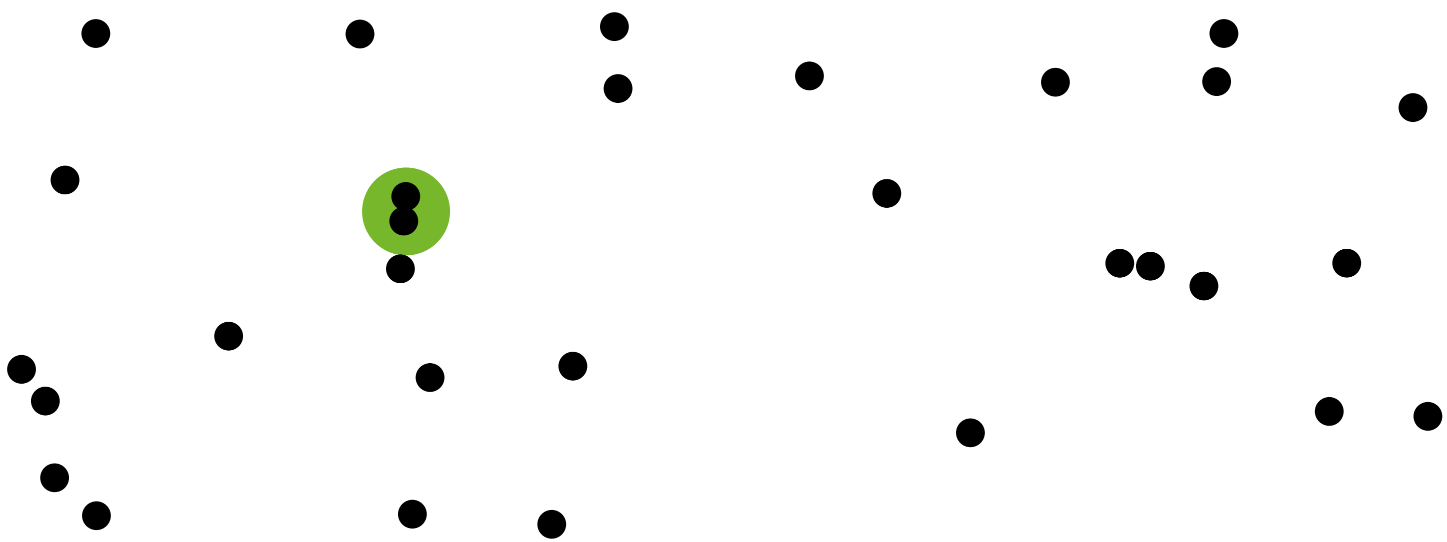


Closest Pair of Points

Problem: Find the pair $\{x, y\} \subset P$ with minimum distance.

- brute-force testing all point pairs $\Rightarrow$ running time $O(n^2)$:-(
- test distances $||\text{rep}(u)\ \text{rep}(v)||$ of all ws-pairs $\{P_u, P_v\}$
  $\Rightarrow$ running time $O(n \log n + s^d n)$:-)

Exercise: For $s > 2$ this actually yields the closest pair.



Discussion

KIT Karlsruhe Institute of Technology

What are further applications of the WSPD?

Discussion

Karlsruhe Institute of Technology

What are further applications of the WSPD?

WSPD is useful whenever one can do without knowing all $\Theta(n^2)$ exact distances in a point set and approximate them instead. One example are force-based layout algorithms in graph drawing, where pairwise repulsive forces of n points need to be calculated.

Discussion

Karlsruhe Institute of Technology

What are further applications of the WSPD?

WSPD is useful whenever one can do without knowing all $\Theta(n^2)$ exact distances in a point set and approximate them instead. One example are force-based layout algorithms in graph drawing, where pairwise repulsive forces of n points need to be calculated.

Why approximate geometrically?

Discussion

Karlsruhe Institute of Technology

What are further applications of the WSPD?

WSPD is useful whenever one can do without knowing all $\Theta(n^2)$ exact distances in a point set and approximate them instead. One example are force-based layout algorithms in graph drawing, where pairwise repulsive forces of n points need to be calculated.

Why approximate geometrically?

On the one hand, this replaces slow computations by faster (but less precise) ones; on the other hand, often the input data are imprecise so that approximate solutions can be sufficient depending on the application.

Discussion

Karlsruhe Institute of Technology

What are further applications of the WSPD?

WSPD is useful whenever one can do without knowing all $\Theta(n^2)$ exact distances in a point set and approximate them instead. One example are force-based layout algorithms in graph drawing, where pairwise repulsive forces of n points need to be calculated.

Why approximate geometrically?

On the one hand, this replaces slow computations by faster (but less precise) ones; on the other hand, often the input data are imprecise so that approximate solutions can be sufficient depending on the application.

Can we achieve the same time bounds with exact computations?

Discussion

What are further applications of the WSPD?

WSPD is useful whenever one can do without knowing all $\Theta(n^2)$ exact distances in a point set and approximate them instead. One example are force-based layout algorithms in graph drawing, where pairwise repulsive forces of n points need to be calculated.

Why approximate geometrically?

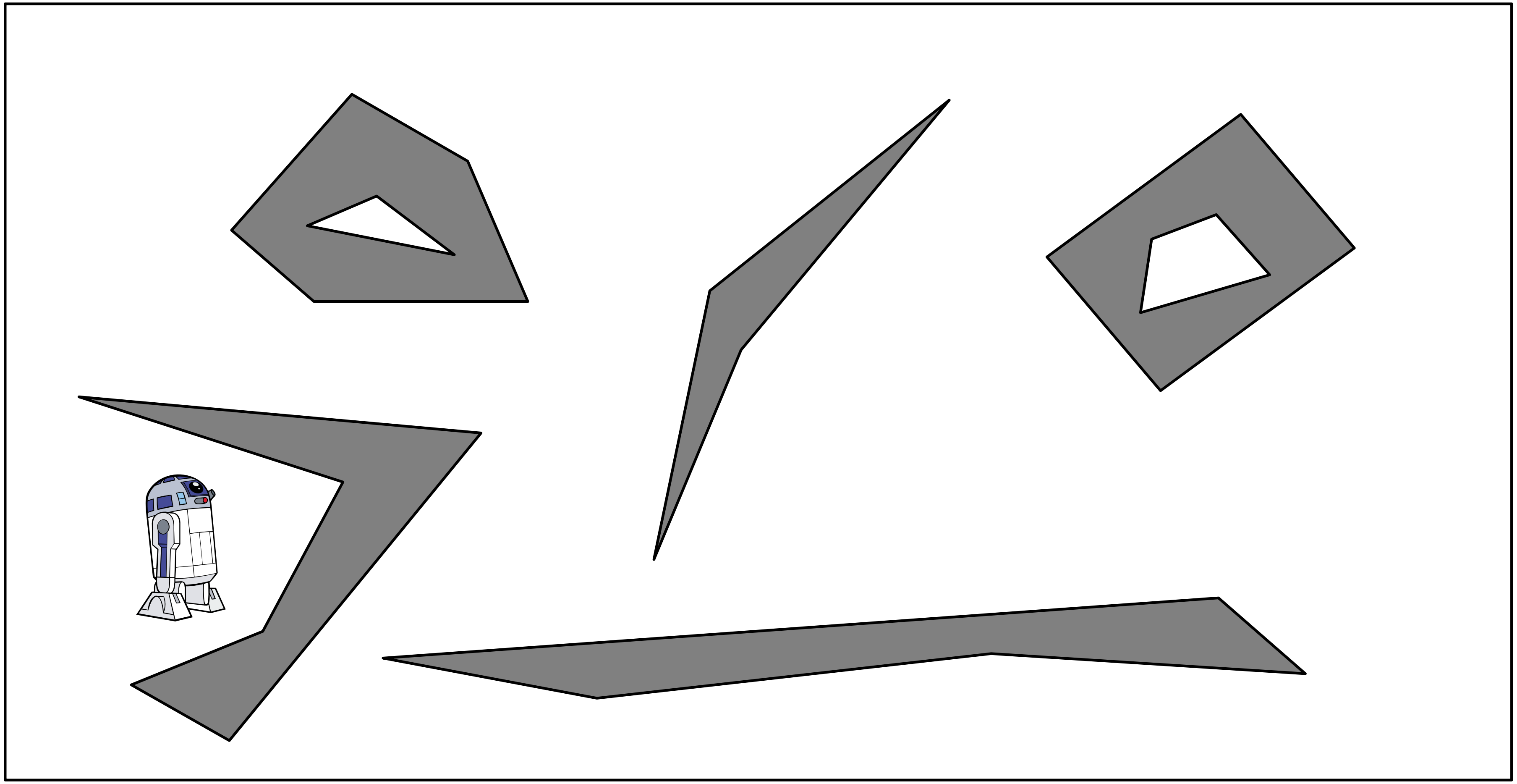
On the one hand, this replaces slow computations by faster (but less precise) ones; on the other hand, often the input data are imprecise so that approximate solutions can be sufficient depending on the application.

Can we achieve the same time bounds with exact computations?

In $\mathbb{R}^2$ this is often true, but not in $\mathbb{R}^d$ for $d > 2$. (e.g. EMST, diameter)

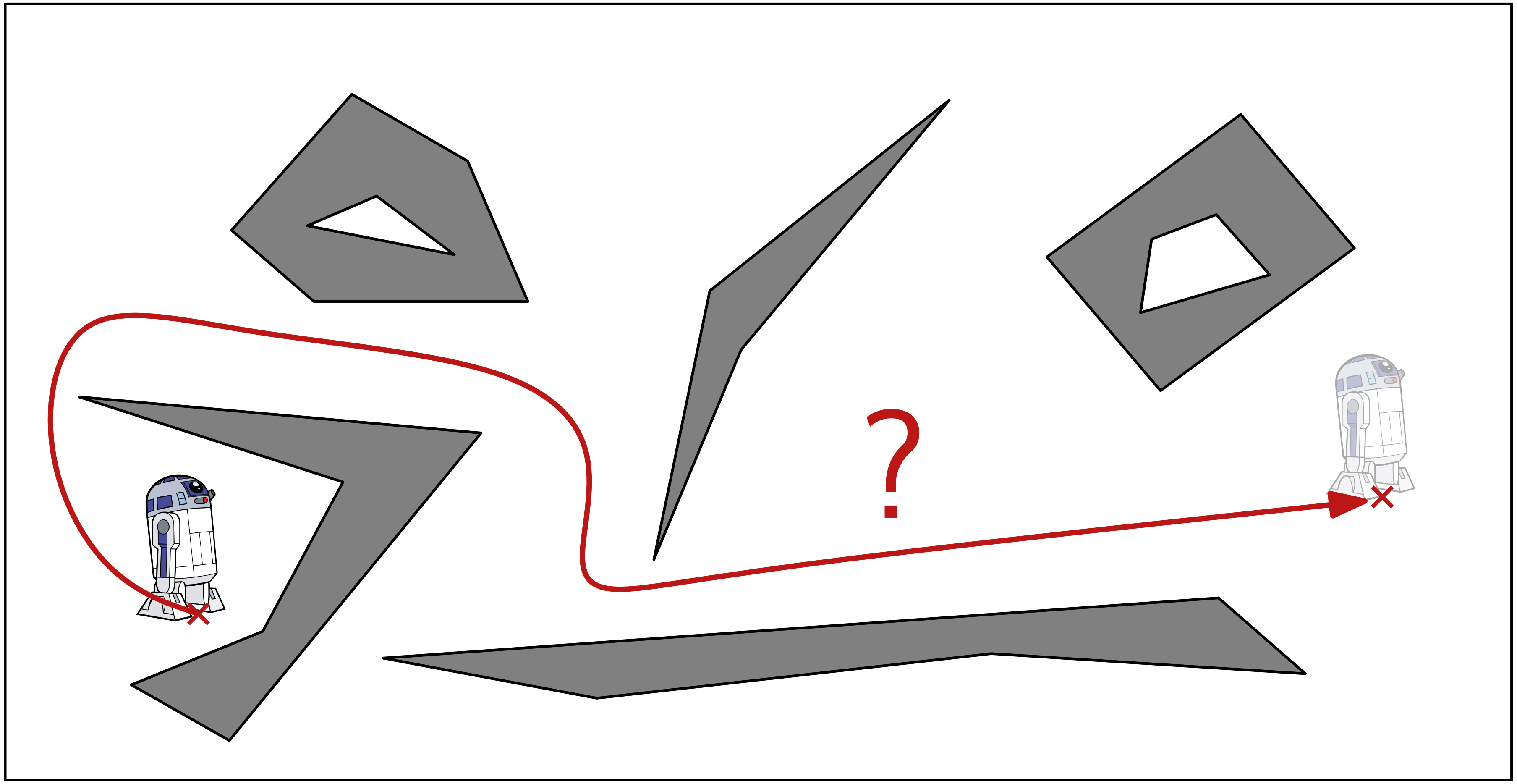
Motion planning and Visibility Graphs

Robot Motion Planning



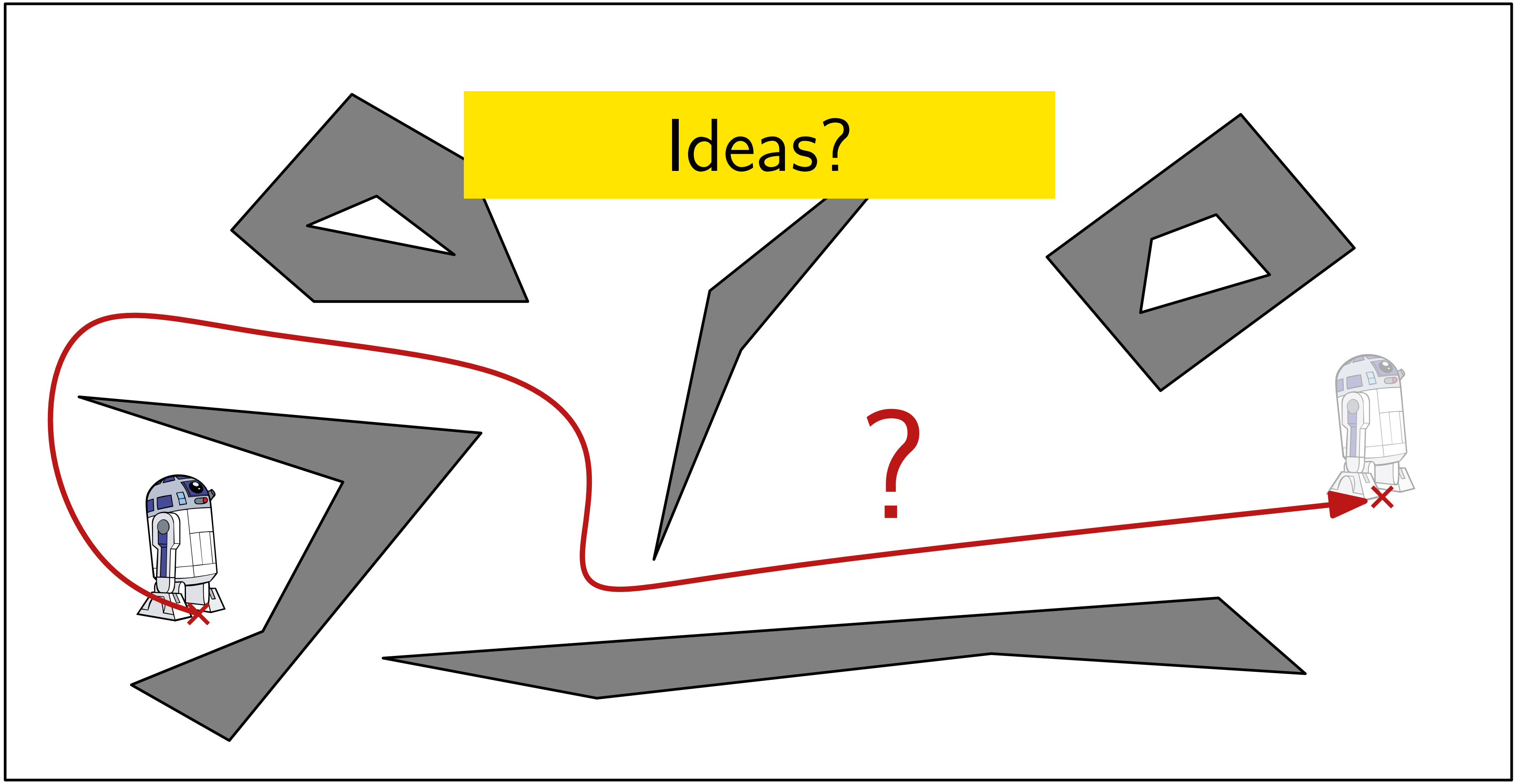
Problem: Given a (point) robot at position p_{start} in a area with polygonal obstacles, find a shortest path to p_{goal} avoiding obstacles.

Robot Motion Planning



Problem: Given a (point) robot at position p_{start} in a area with polygonal obstacles, find a shortest path to p_{goal} avoiding obstacles.

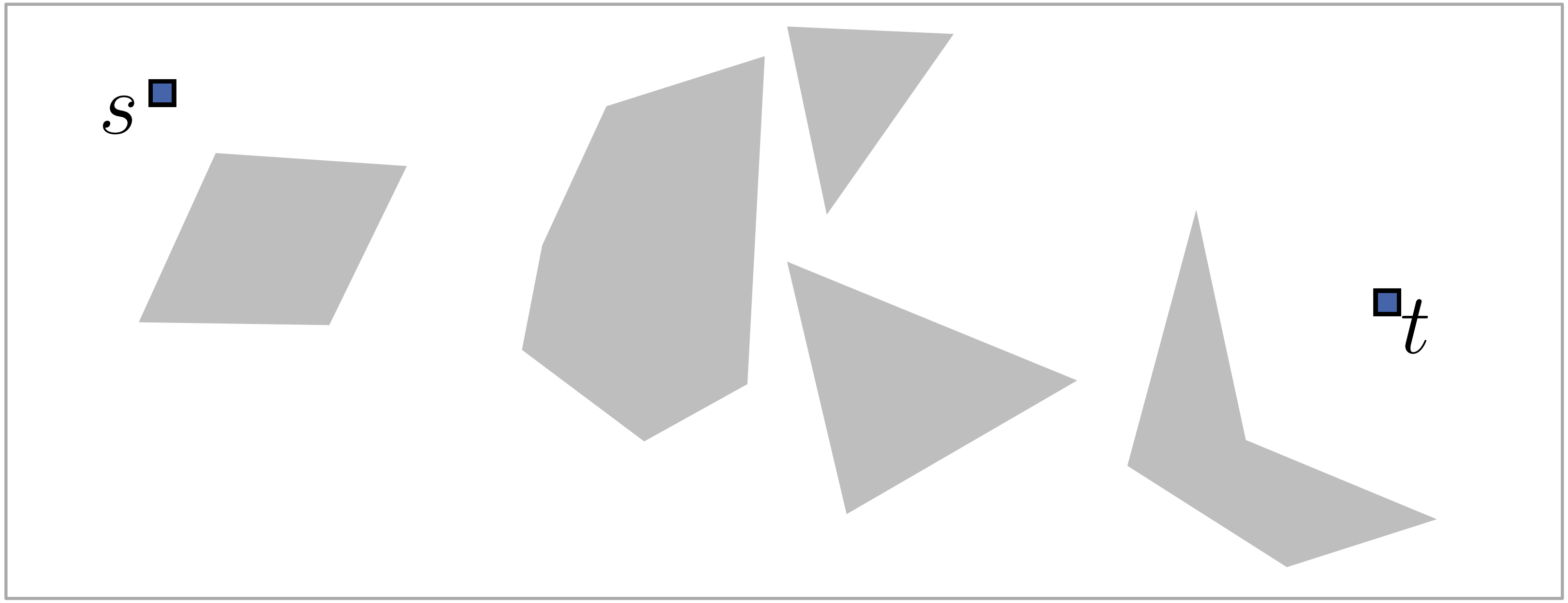
Robot Motion Planning



Problem: Given a (point) robot at position p_{start} in a area with polygonal obstacles, find a shortest path to p_{goal} avoiding obstacles.

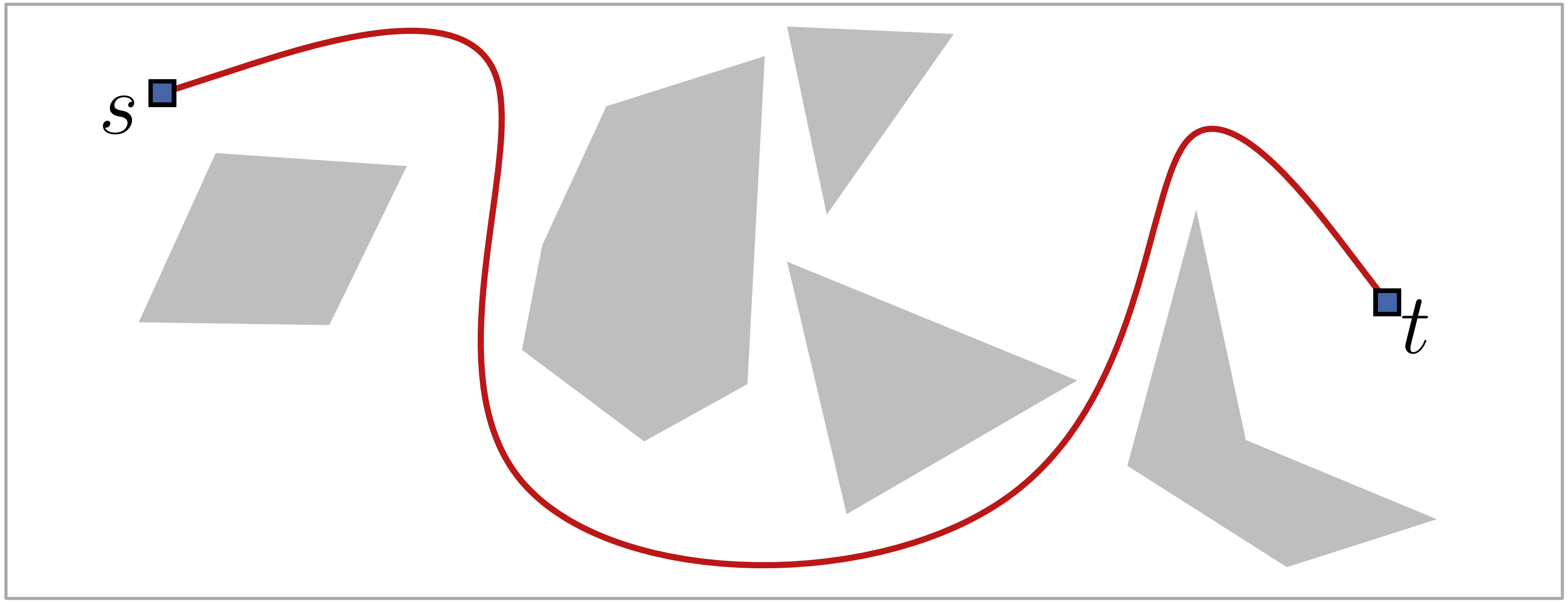
Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S



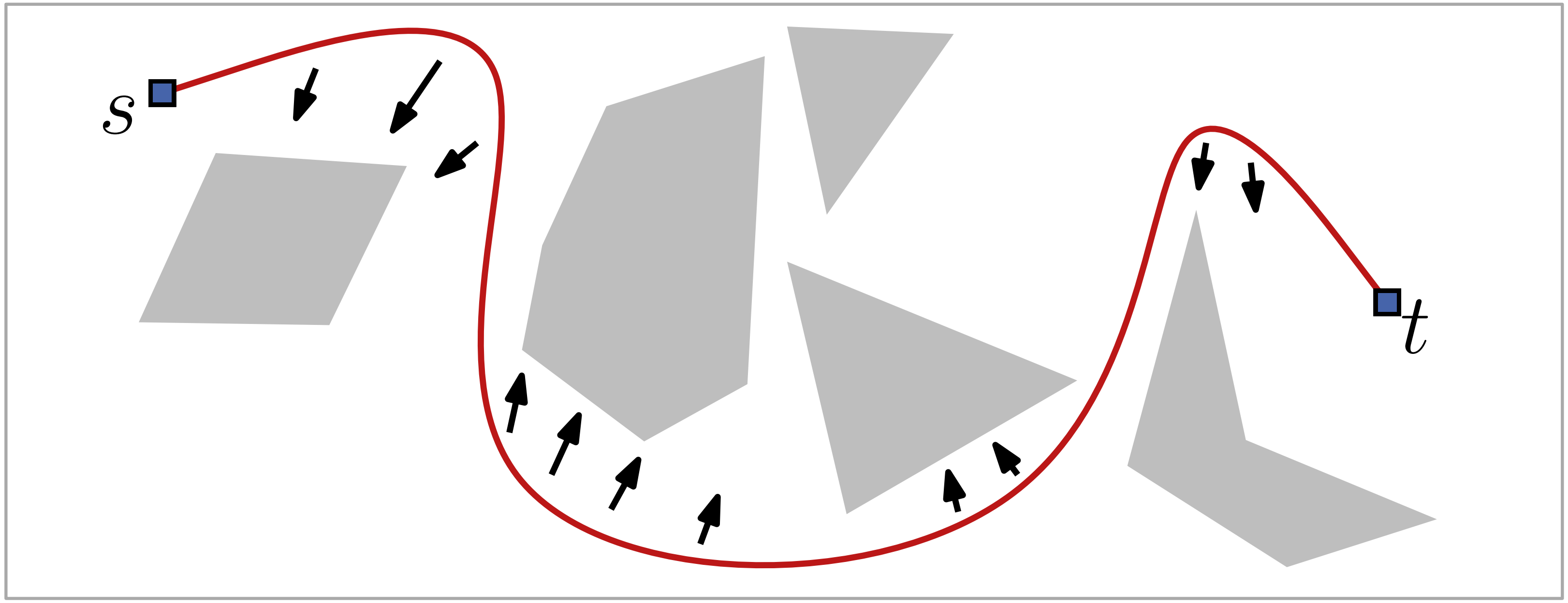
Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



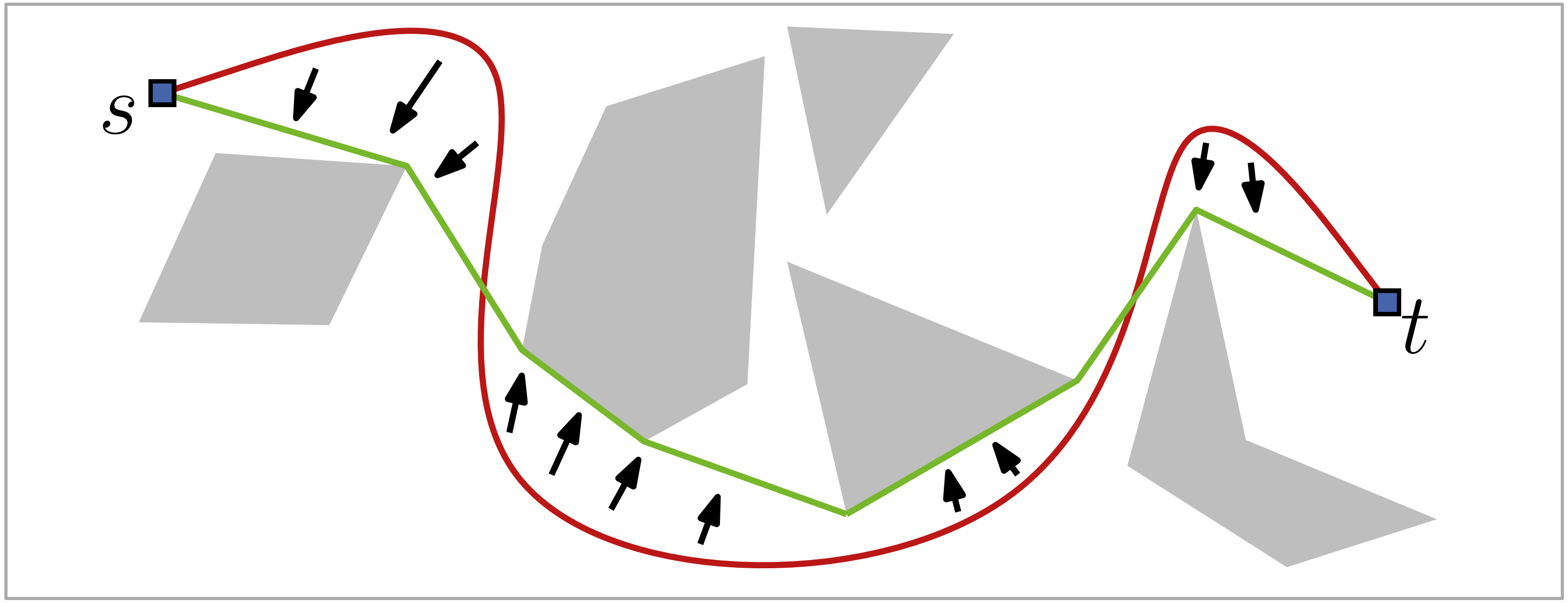
Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



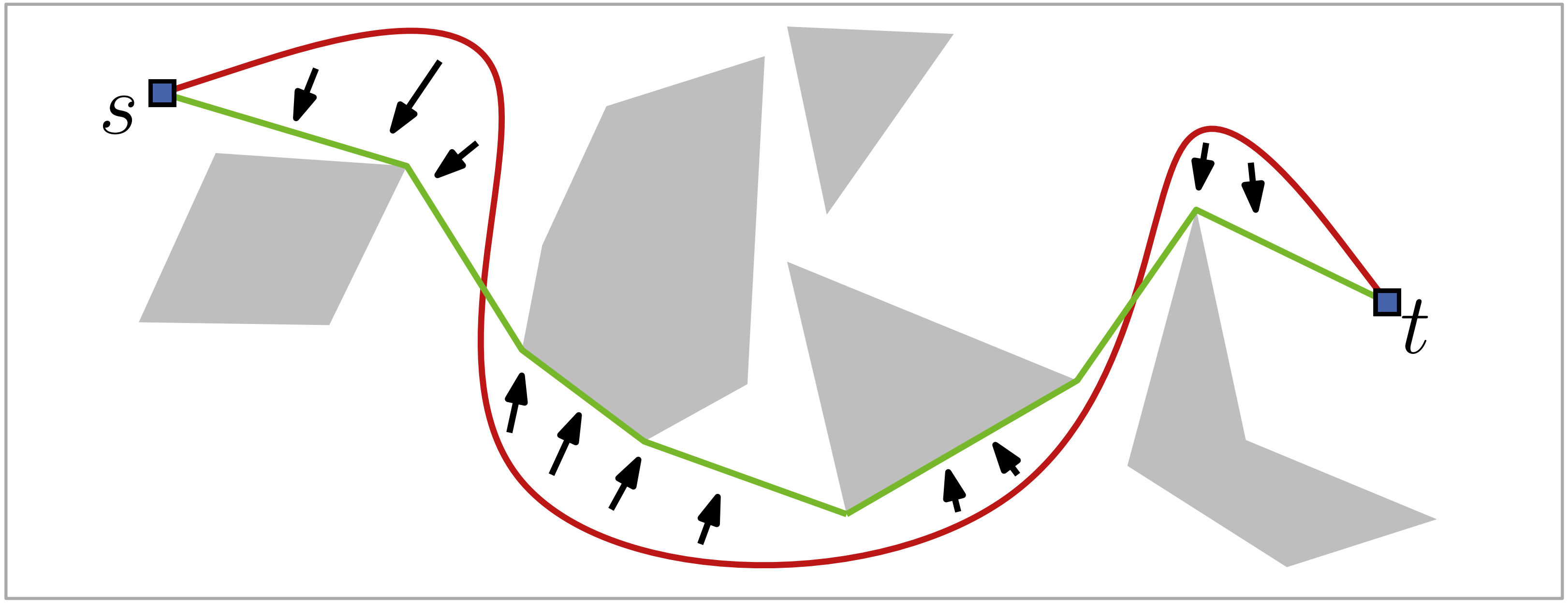
Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



Shortest Paths in Polygonal Areas

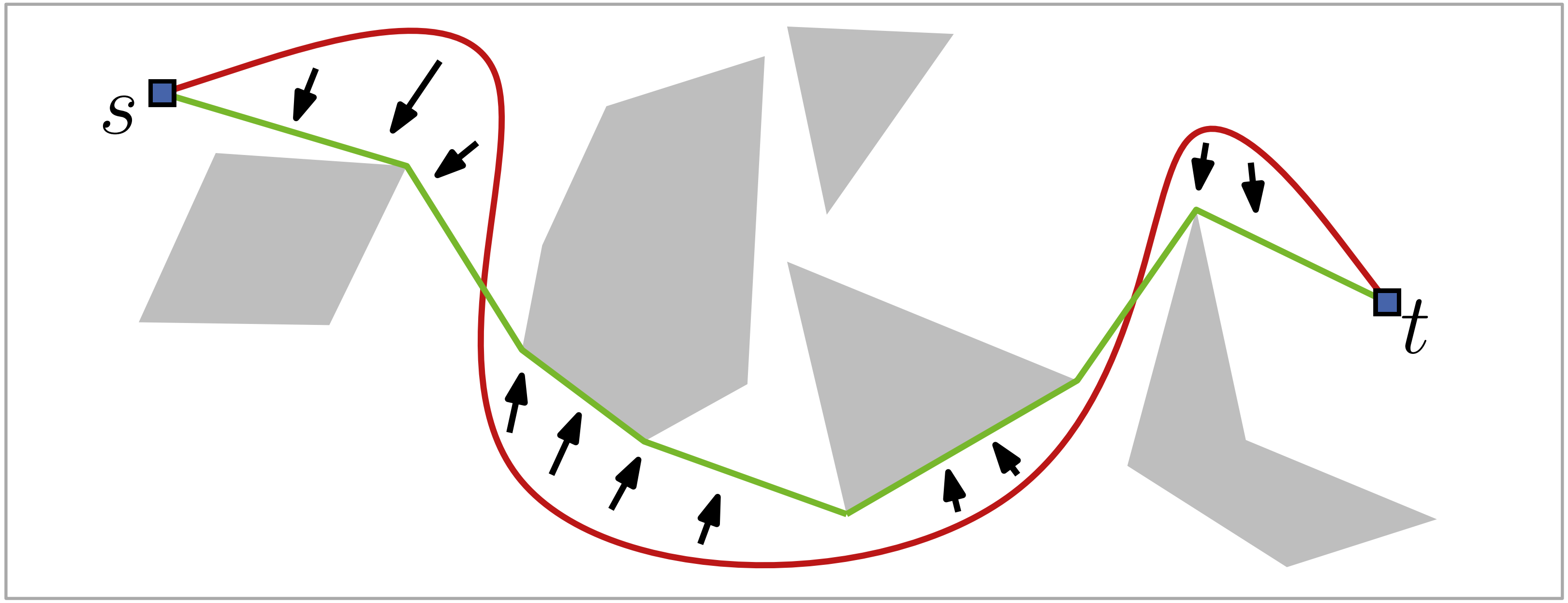
Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



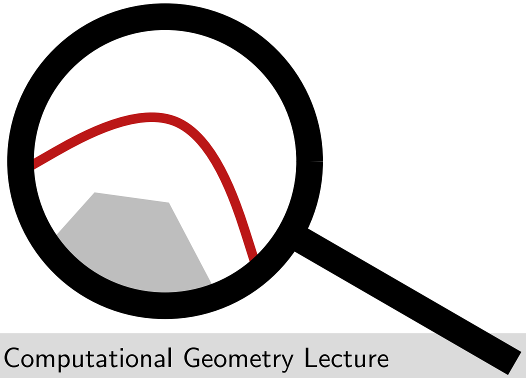
Proof sketch:

Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

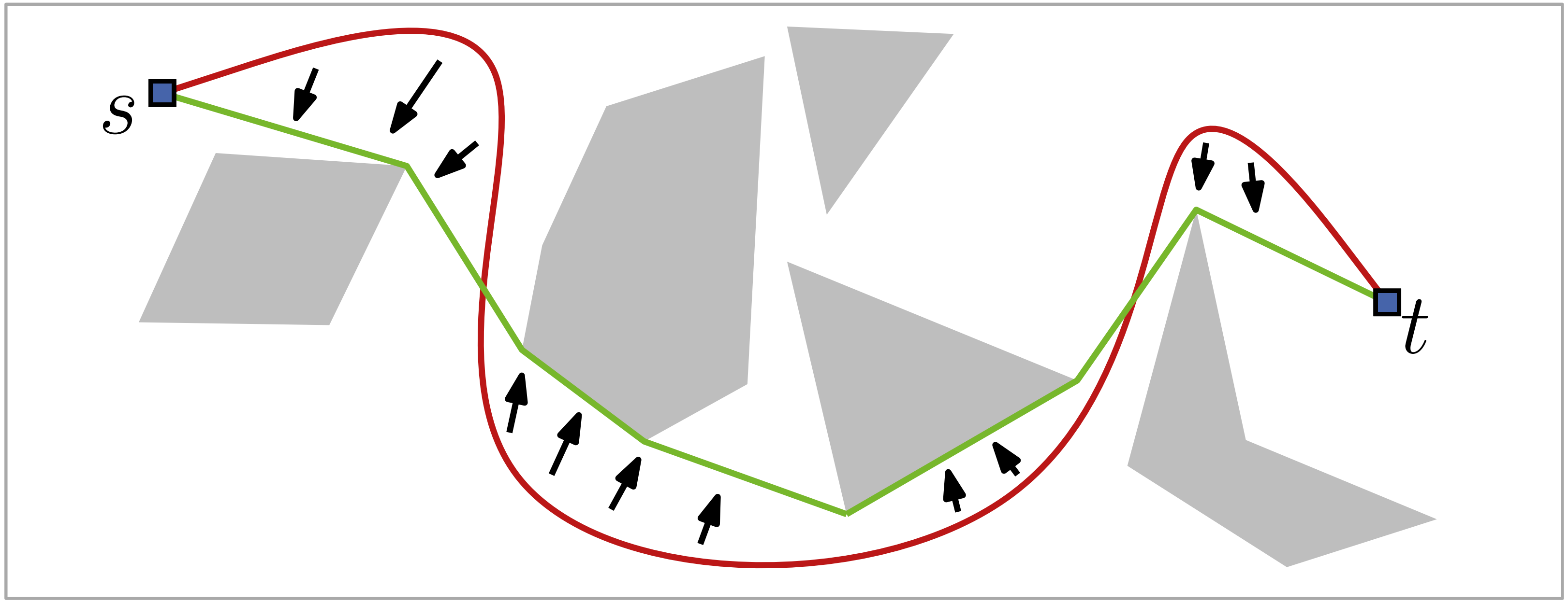


Proof sketch:

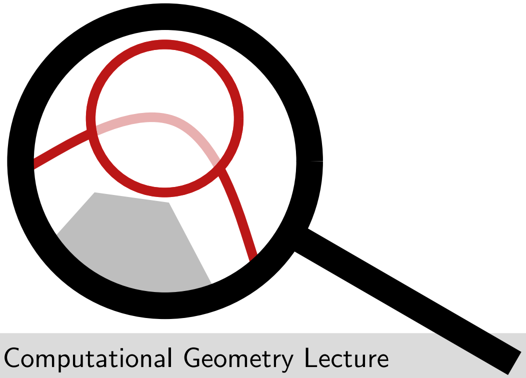


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

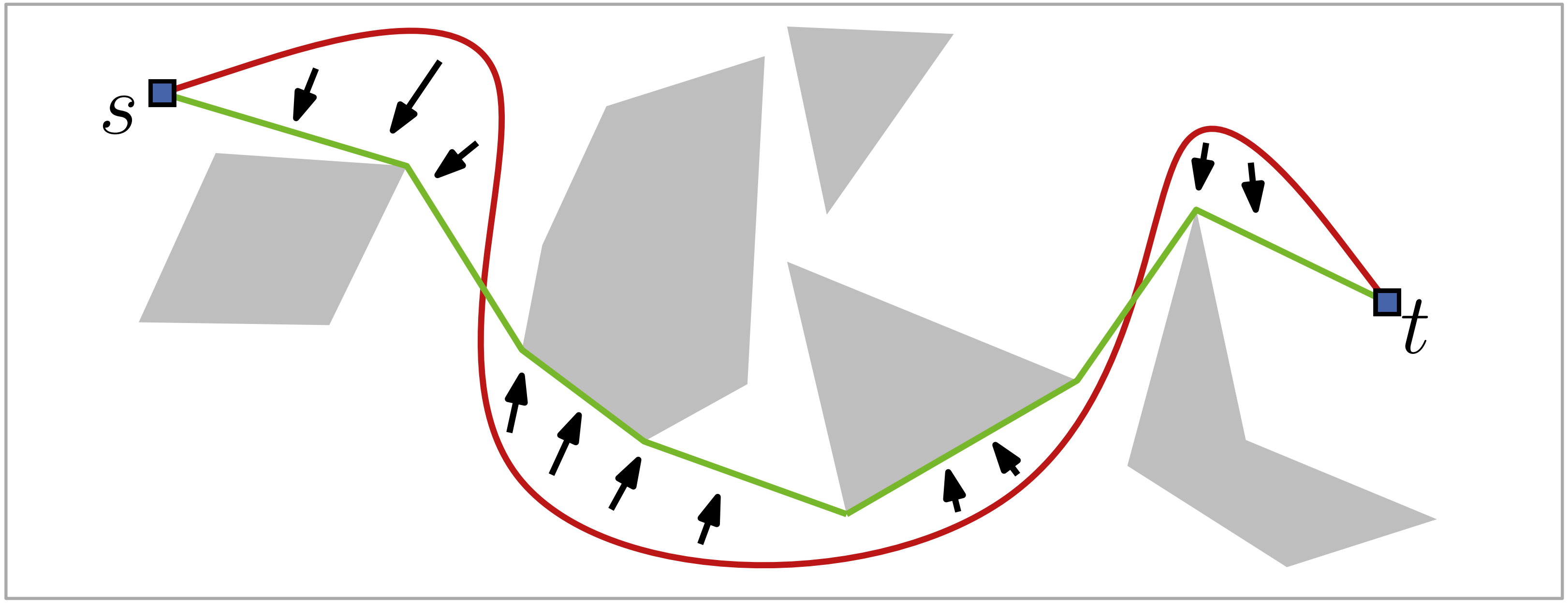


Proof sketch:

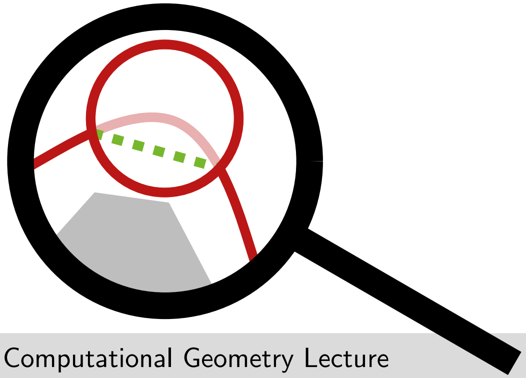


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

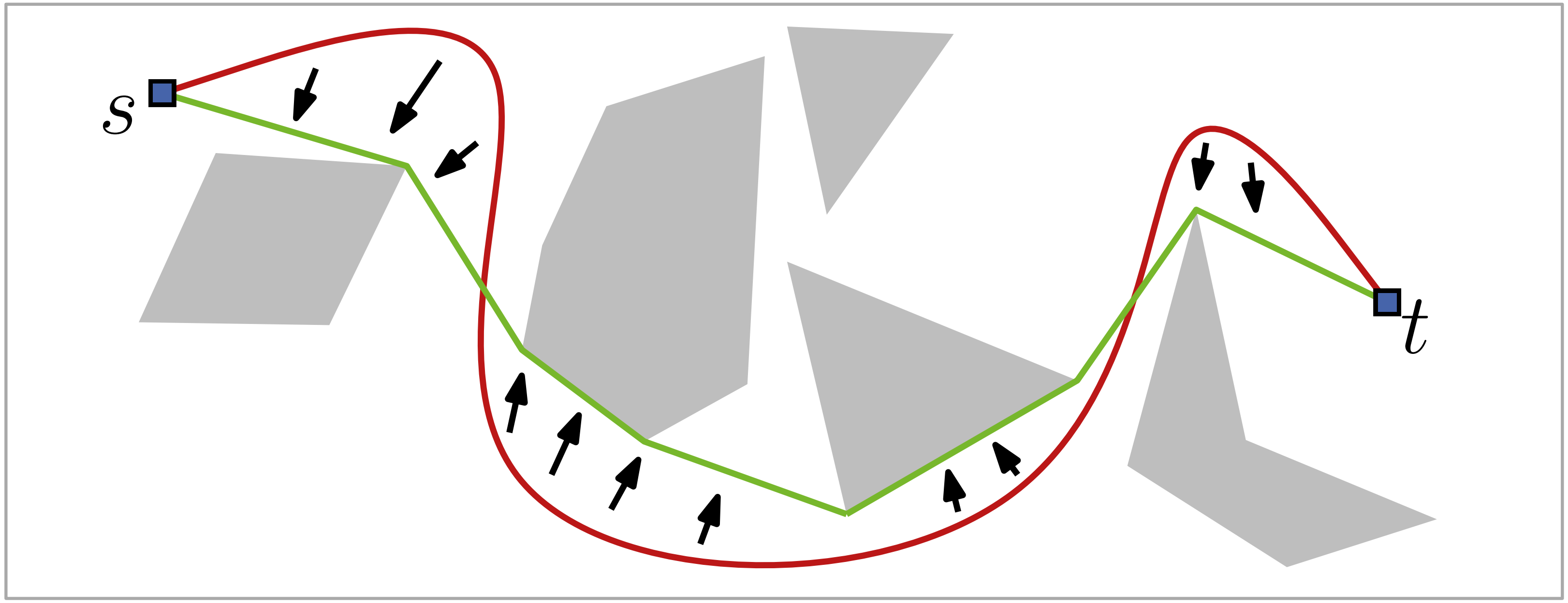


Proof sketch:

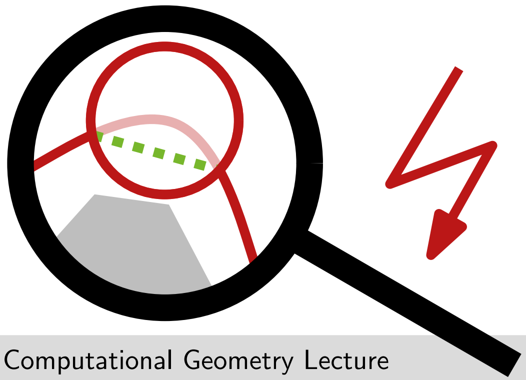


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

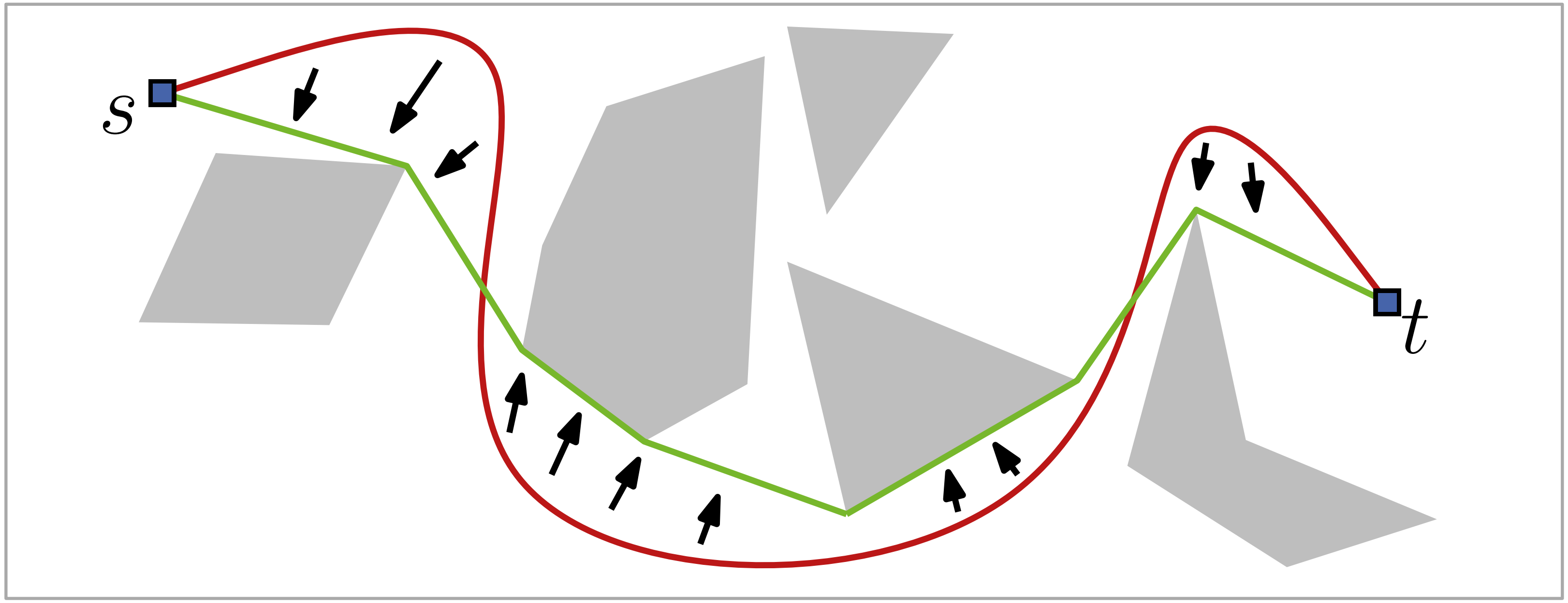


Proof sketch:

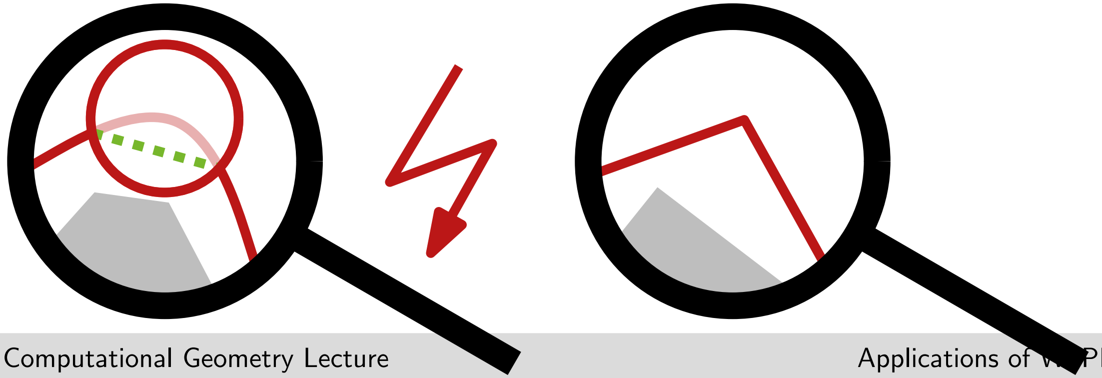


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

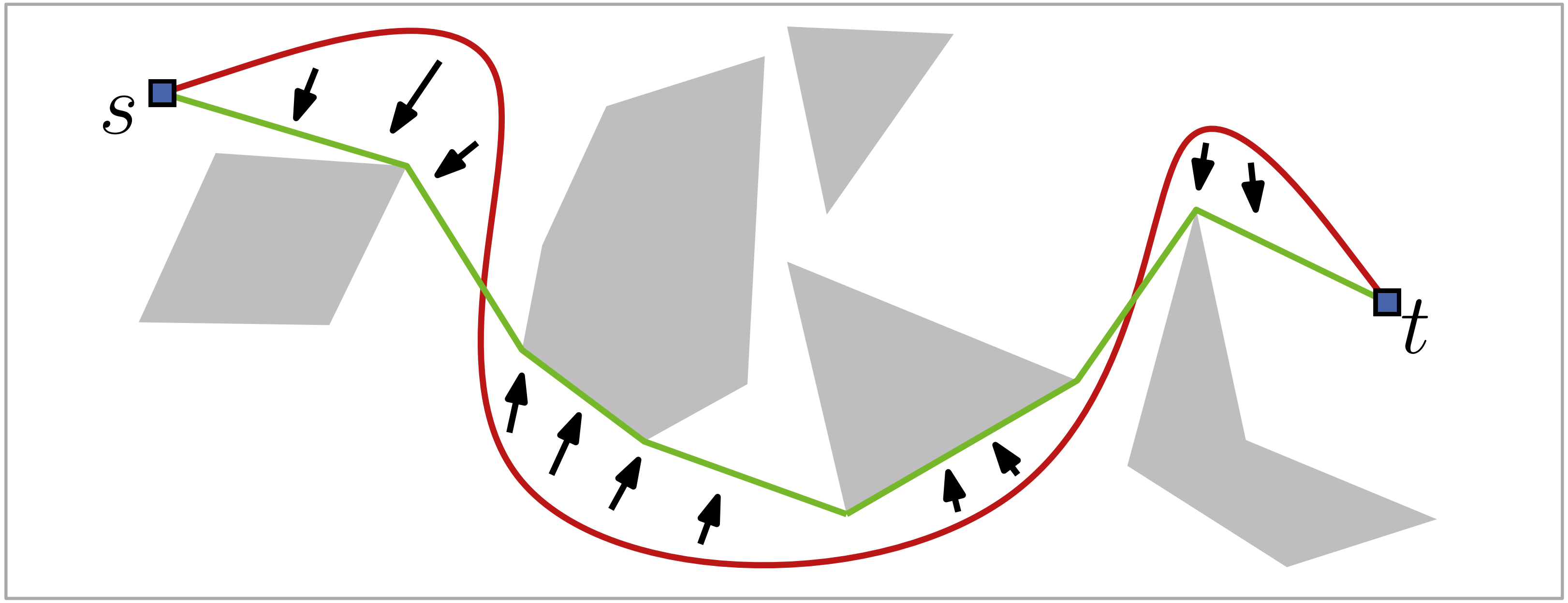


Proof sketch:

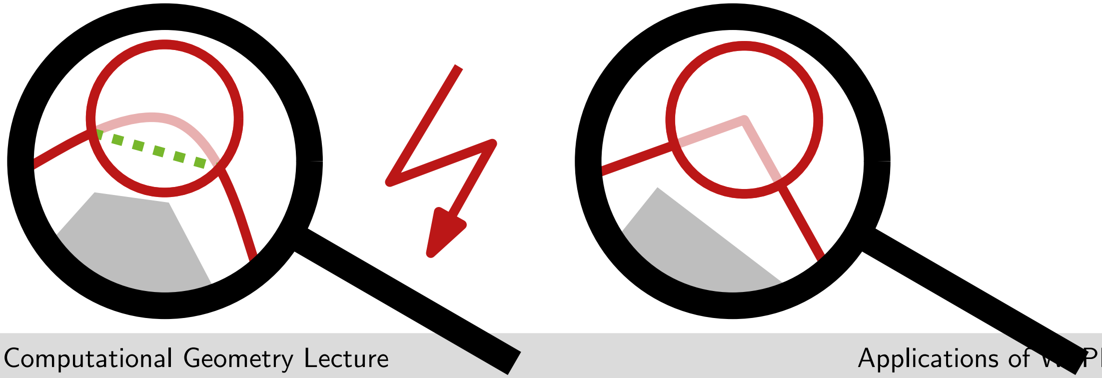


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



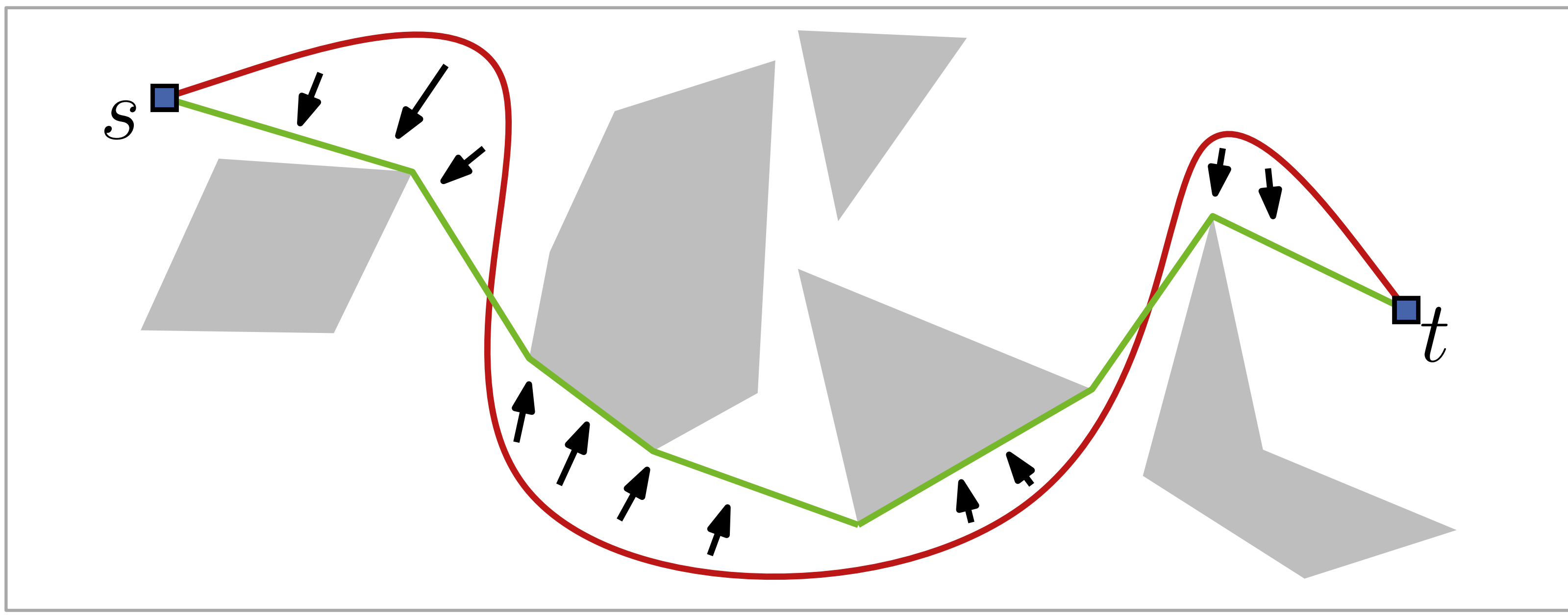
Proof sketch:



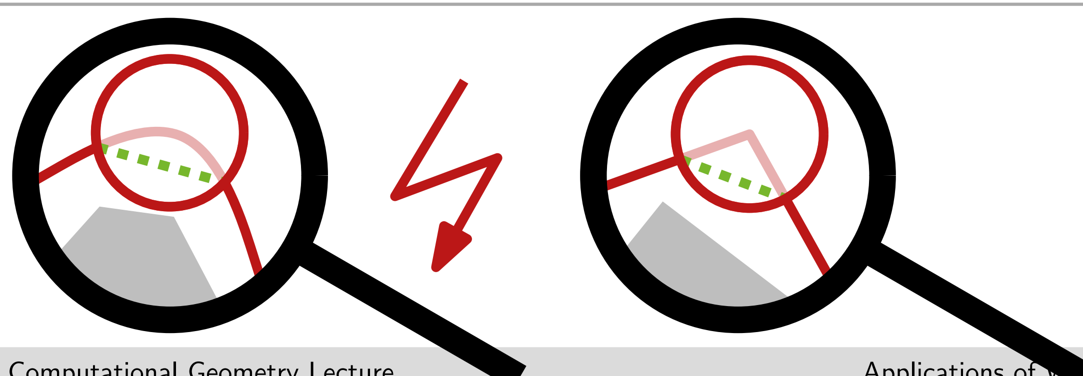
Shortest Paths in Polygonal Areas

Karlsruhe Institute of Technology

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

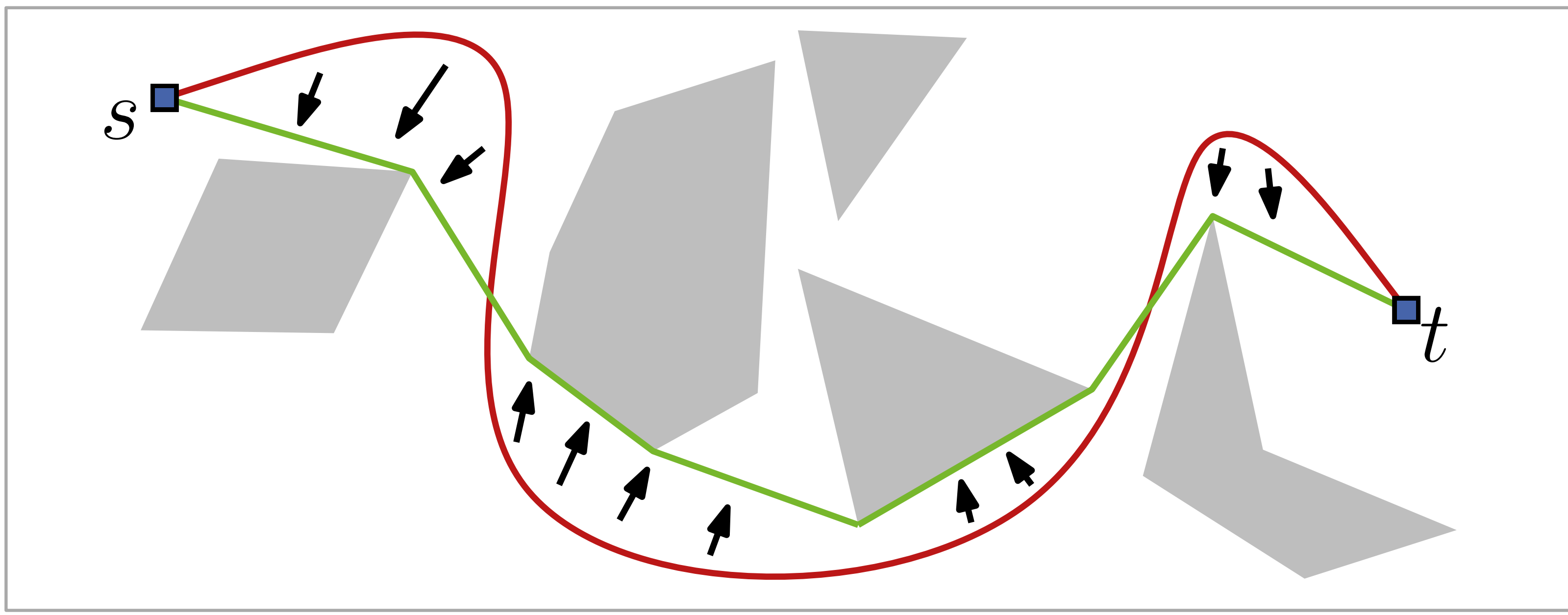


Proof sketch:

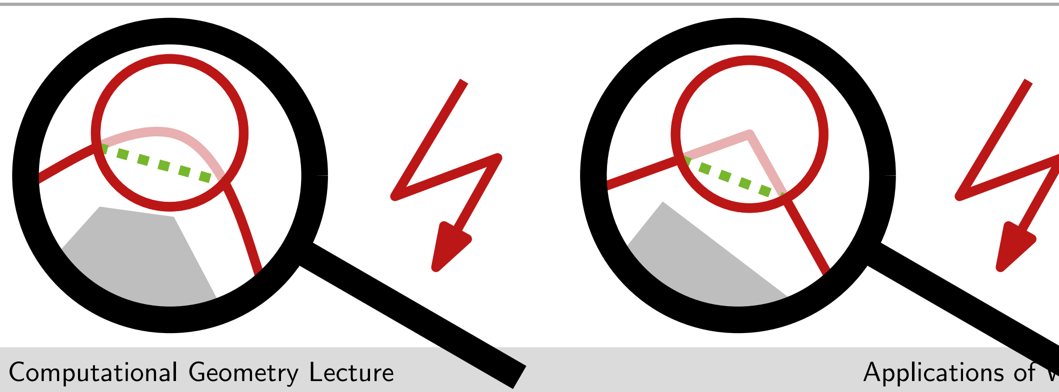


Shortest Paths in Polygonal Areas

Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.

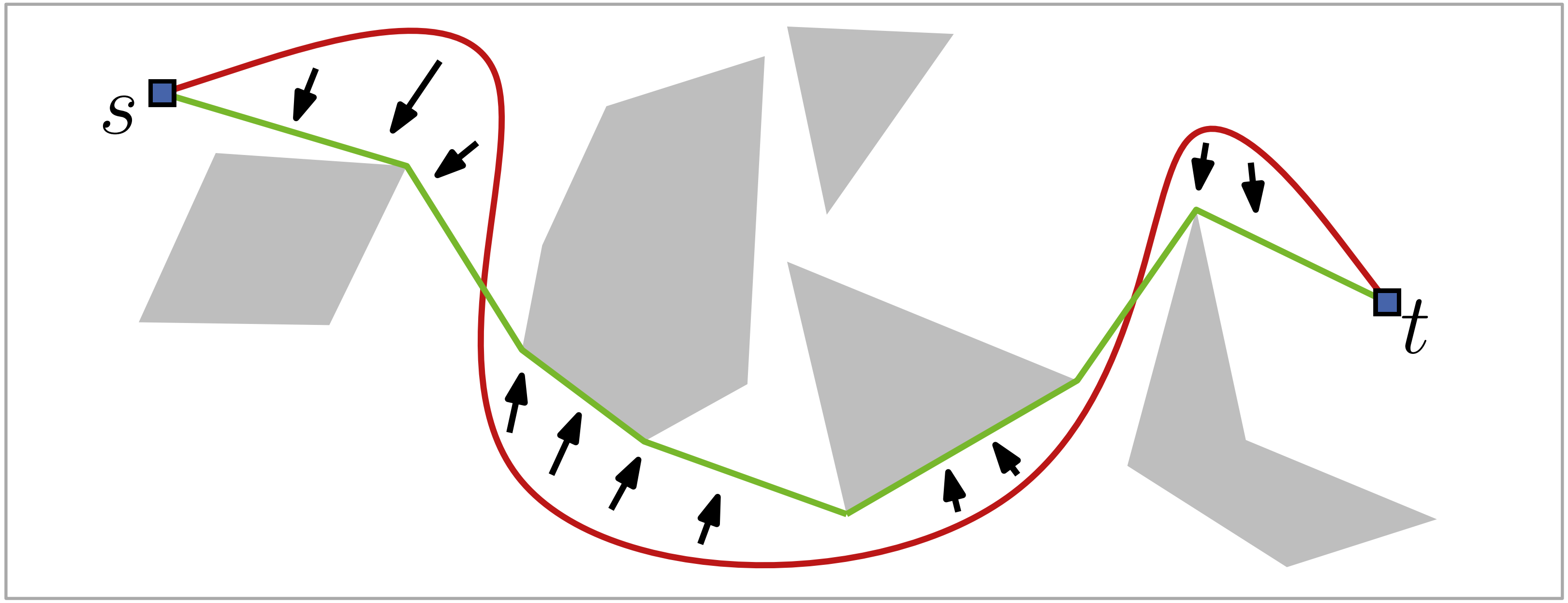


Proof sketch:



Shortest Paths in Polygonal Areas

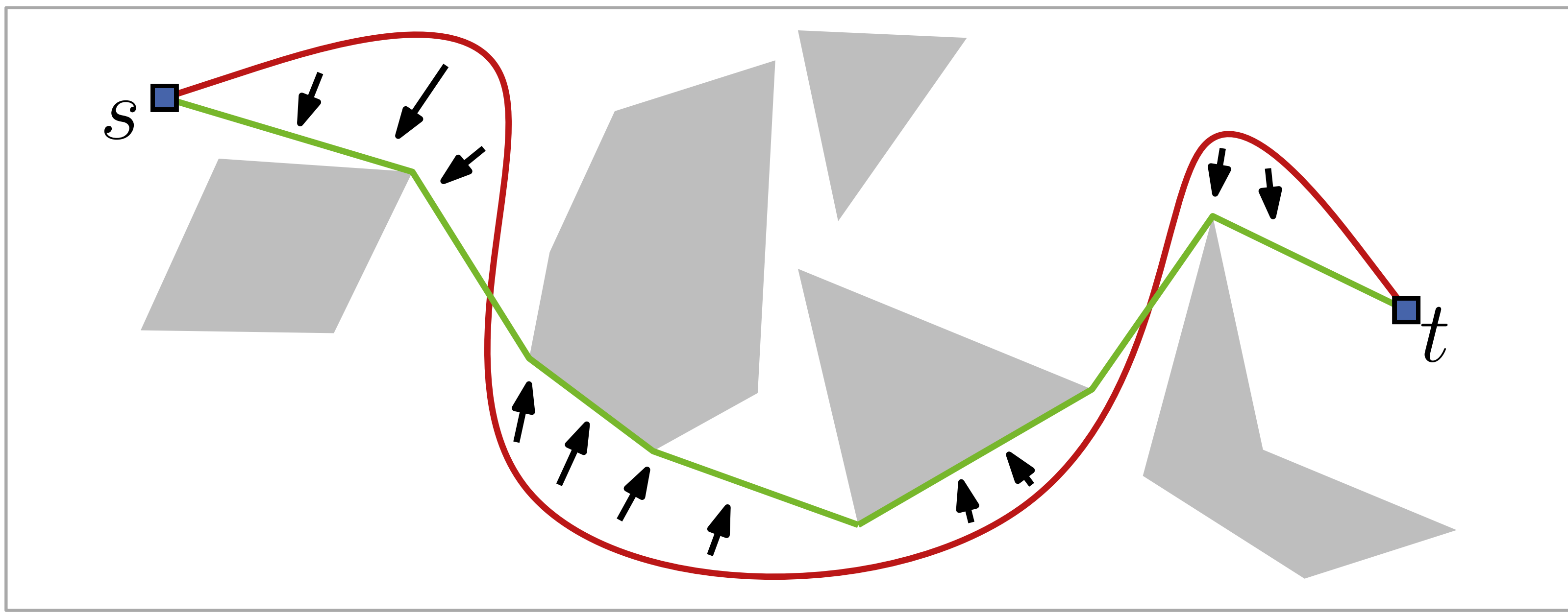
Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S
each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



Proof sketch:

Shortest Paths in Polygonal Areas

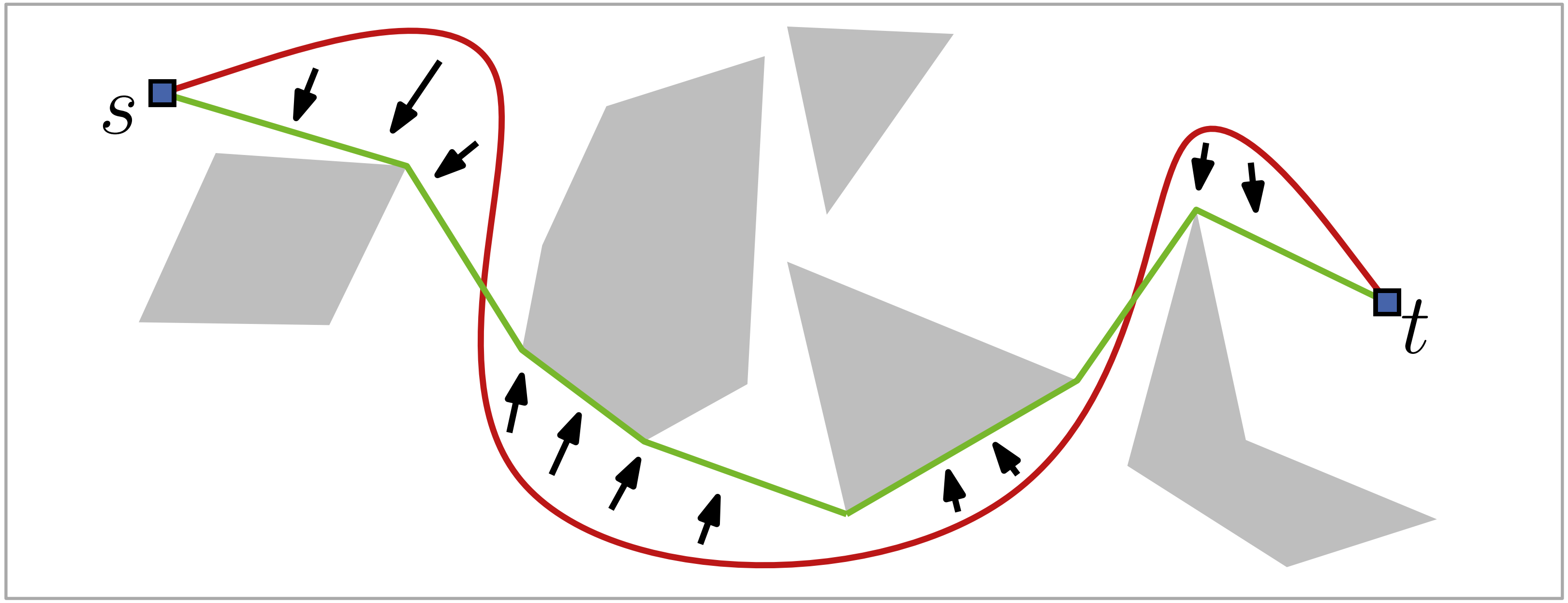
Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



Proof sketch:

Shortest Paths in Polygonal Areas

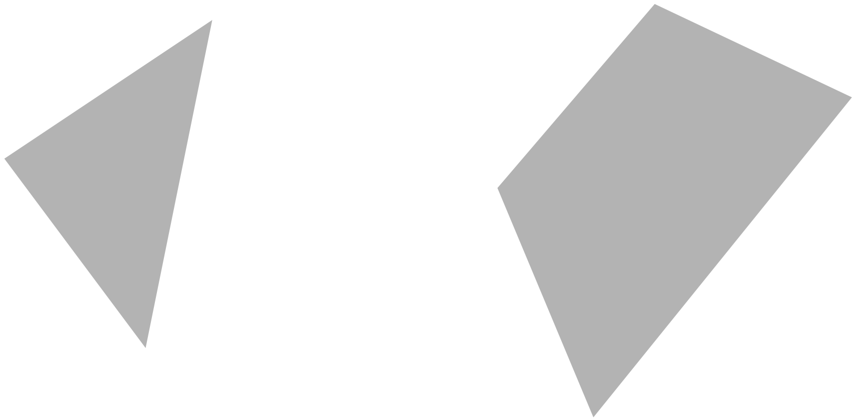
Lemma 1: For a set S of disjoint open polygons in $\mathbb{R}^2$ and two points s and t not in S each shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal path whose internal vertices are vertices of S.



Proof sketch:

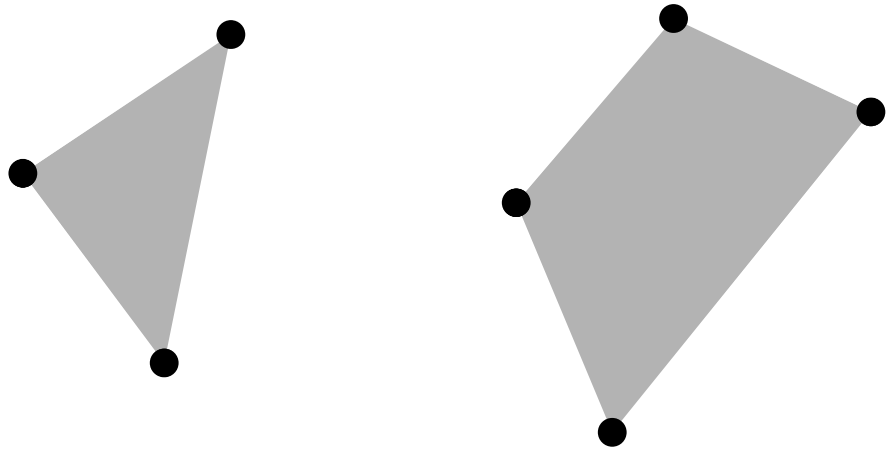
Visibility Graph

Given a set S of disjoint open polygons...



Visibility Graph

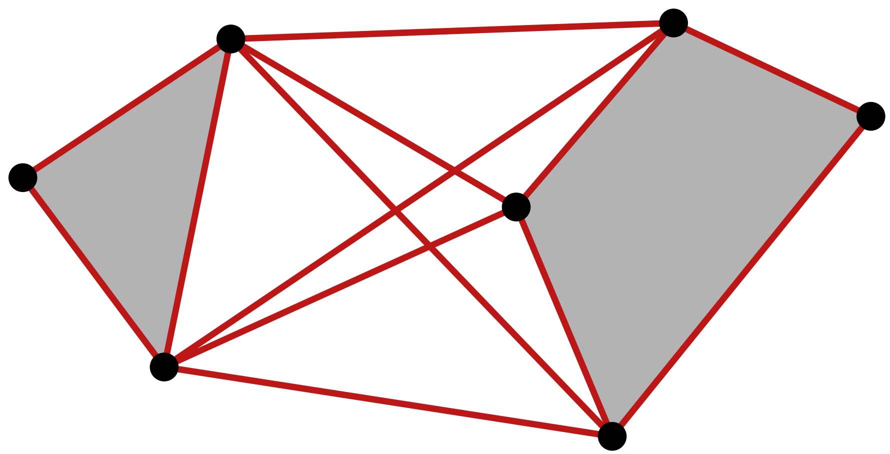
Given a set S of disjoint open polygons...



...with point set $V(S)$.

Visibility Graph

Given a set S of disjoint open polygons...



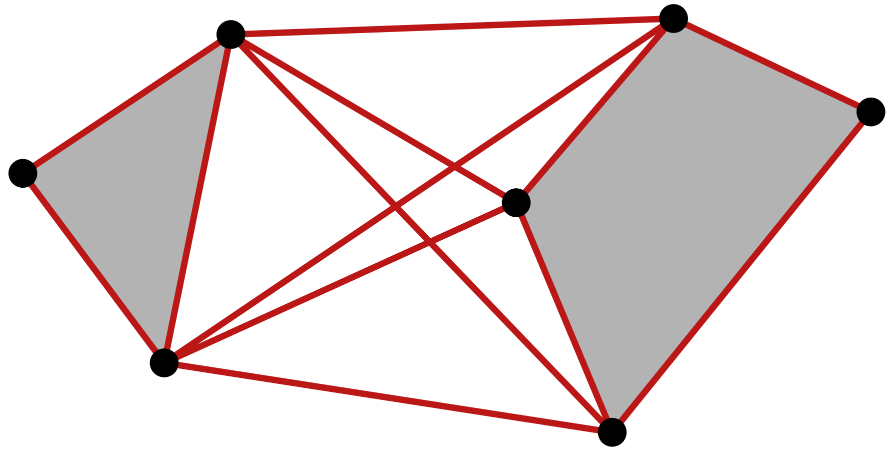
...with point set $V(S)$.

Def.: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the **visibility graph** of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S) \text{ and } u \textit{ sees } v\}$ und $w(uv) = |uv|$.

Visibility Graph

Karlsruhe Institute of Technology

Given a set S of disjoint open polygons...

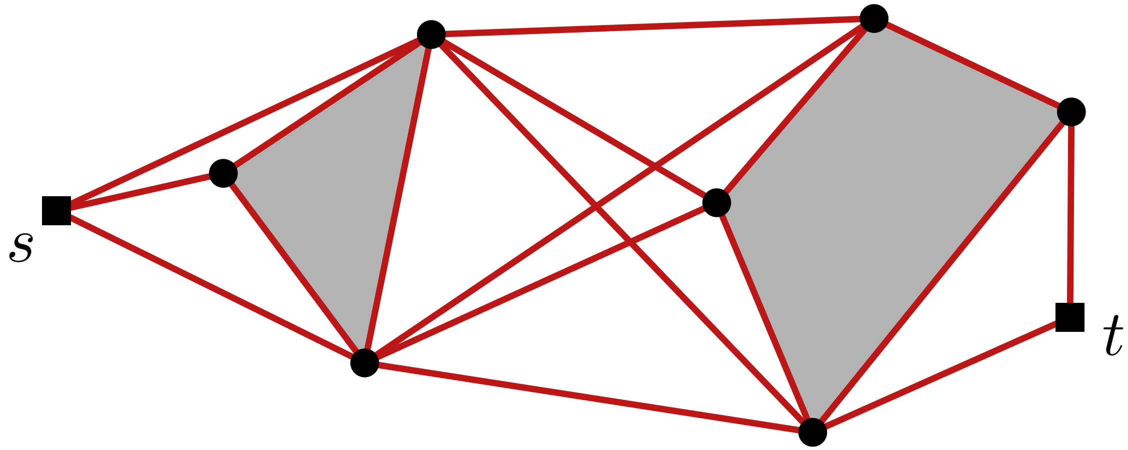


...with point set $V(S)$.

Def.: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the **visibility graph** of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S)$ and u *sees* $v\}$ und $w(uv) = |uv|$.
Where u **sees** v $:\Leftrightarrow \overline{uv} \cap \bigcup S = \emptyset$

Visibility Graph

Given a set S of disjoint open polygons...



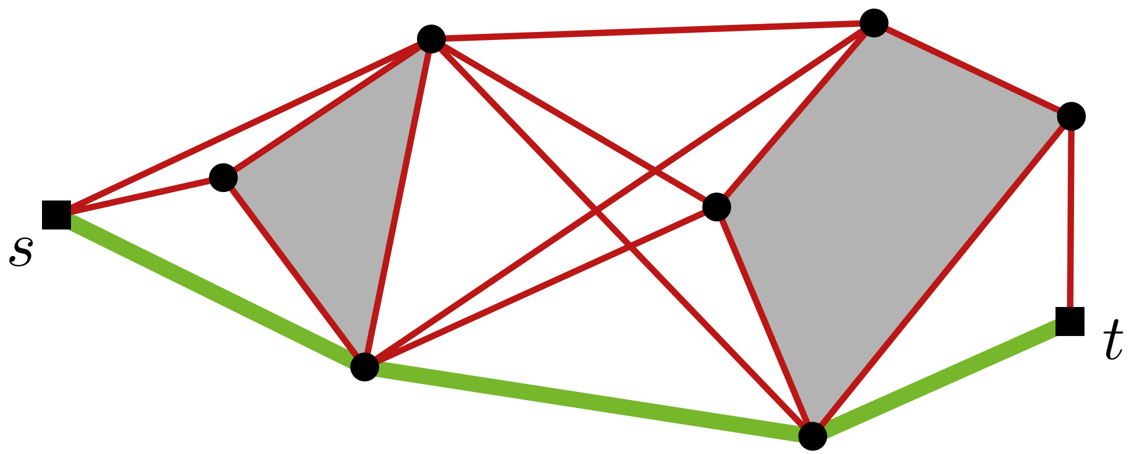
...with point set $V(S)$.

Def.: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the **visibility graph** of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S)$ and u *sees* $v\}$ und $w(uv) = |uv|$.
Where u **sees** $v :\Leftrightarrow \overline{uv} \cap \bigcup S = \emptyset$

Define $S^\star = S \cup \{s, t\}$ and $G_{\text{vis}}(S^\star)$ analogously.

Visibility Graph

Given a set S of disjoint open polygons...



...with point set $V(S)$.

Def.: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the **visibility graph** of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S) \text{ and } u \textit{ sees } v\}$ und $w(uv) = |uv|$.
Where u **sees** v $:\Leftrightarrow \overline{uv} \cap \bigcup S = \emptyset$

Define $S^\star = S \cup \{s, t\}$ and $G_{\text{vis}}(S^\star)$ analogously.

Lemma 1
$\Rightarrow$

A shortest st-path in $\mathbb{R}^2$ avoiding obstacles in S is equivalent to a shortest st-path in $G_{\text{vis}}(S^\star)$.

Algorithm

ShortestPath(S, s, t)

Input: Obstacles S, points $s, t \in \mathbb{R}^2 \setminus \bigcup S$
Output: Shortest collision-free st-path in S

1 $G_{\mathsf{vis}} \leftarrow$ VisibilityGraph$(S \cup \{s, t\})$
2 **foreach** $uv \in E_{\mathsf{vis}}$ **do** $w(uv) \leftarrow |uv|$
3 **return** Dijkstra$(G_{\mathsf{vis}}, w, s, t)$

Algorithm

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

Input: Obstacles S, points $s, t \in \mathbb{R}^2 \setminus \bigcup S$
Output: Shortest collision-free st-path in S

```
G_vis ← VisibilityGraph(S ∪ {s, t})                      ?
foreach uv ∈ E_vis do w(uv) ← |uv|                       O(m)
return Dijkstra(G_vis, w, s, t)                          O(n log n + m)
```

Algorithm

Karlsruhe Institute of Technology

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

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

Output: Shortest collision-free st-path in S

1 $G_{\text{vis}} \leftarrow$ VisibilityGraph$(S \cup \{s, t\})$ $\quad O(n^2 \log n)$

2 **foreach** $uv \in E_{\text{vis}}$ **do** $w(uv) \leftarrow |uv|$ $\quad O(m)$

3 **return** Dijkstra$(G_{\text{vis}}, w, s, t)$ $\quad O(n \log n + m)$

$O(n^2 \log n)$

Thm 1: A shortest st-path in an area with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Computing a Visibility Graph

VisibilityGraph(S)

Input: Set of disjoint polygons S

Output: Visibility graph $G_{\mathsf{vis}}(S)$

```
E ← ∅
foreach v ∈ V(S) do
    W ← VisibleVertices(v, S)
    E ← E ∪ {vw | w ∈ W}
return (V(S), E)
```

$E \leftarrow \emptyset$

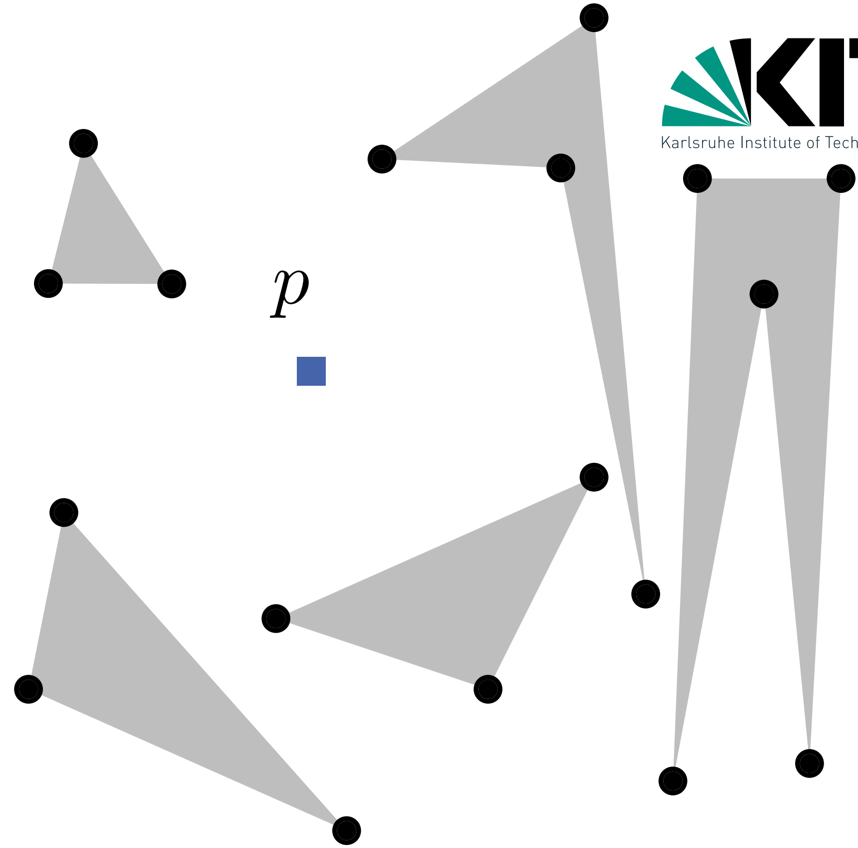
foreach $v \in V(S)$ **do**

- $W \leftarrow \text{VisibleVertices}(v, S)$
- $E \leftarrow E \cup \{vw \mid w \in W\}$

return $(V(S), E)$

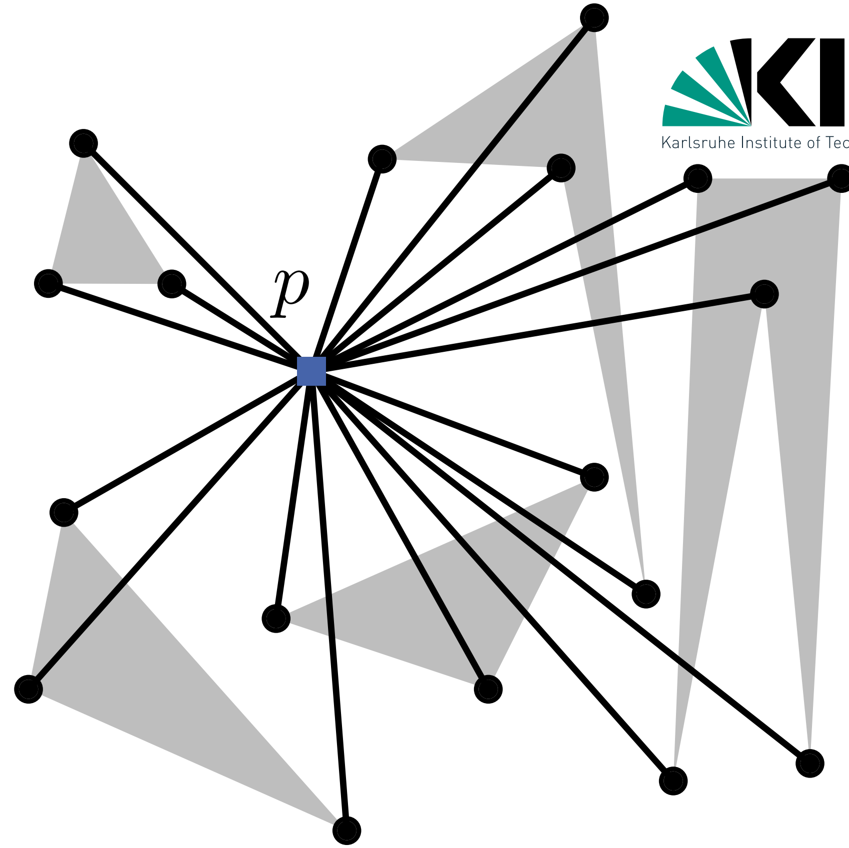
Computing Visible Nodes

VisibleVertices(p, S)



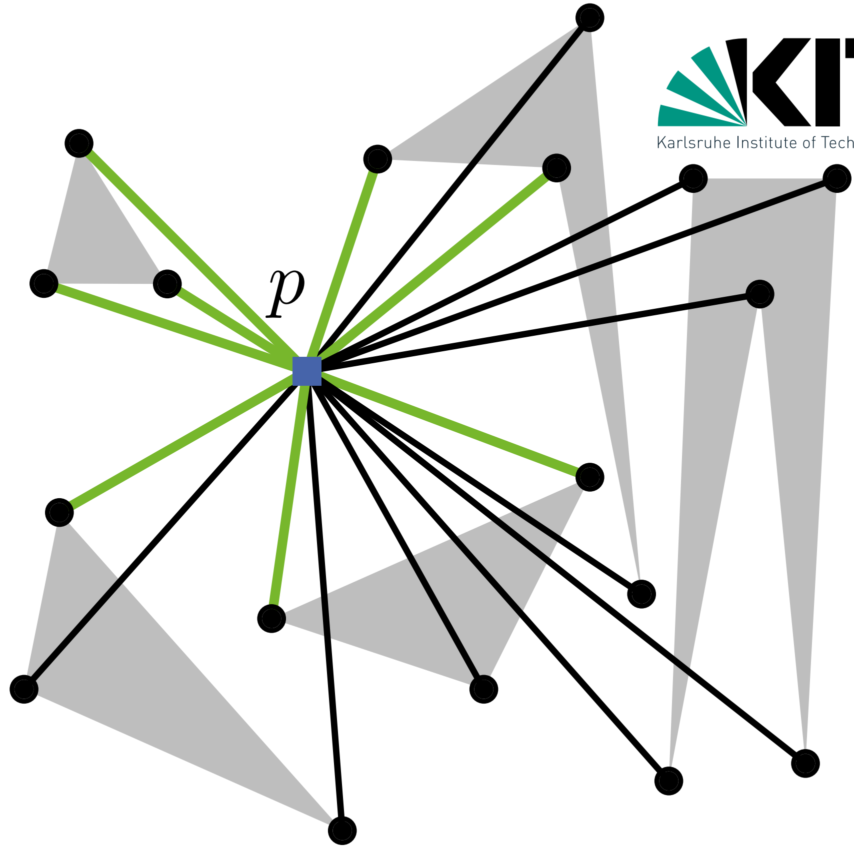
Computing Visible Nodes

VisibleVertices(p, S)



Computing Visible Nodes

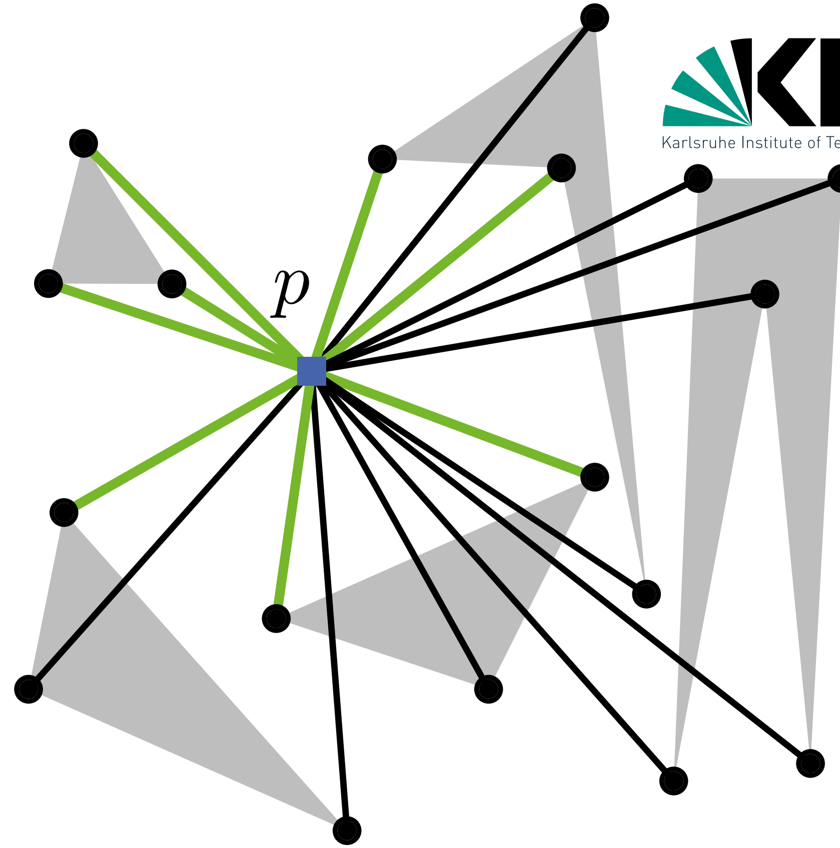
VisibleVertices(p, S)



Computing Visible Nodes

VisibleVertices(p, S)

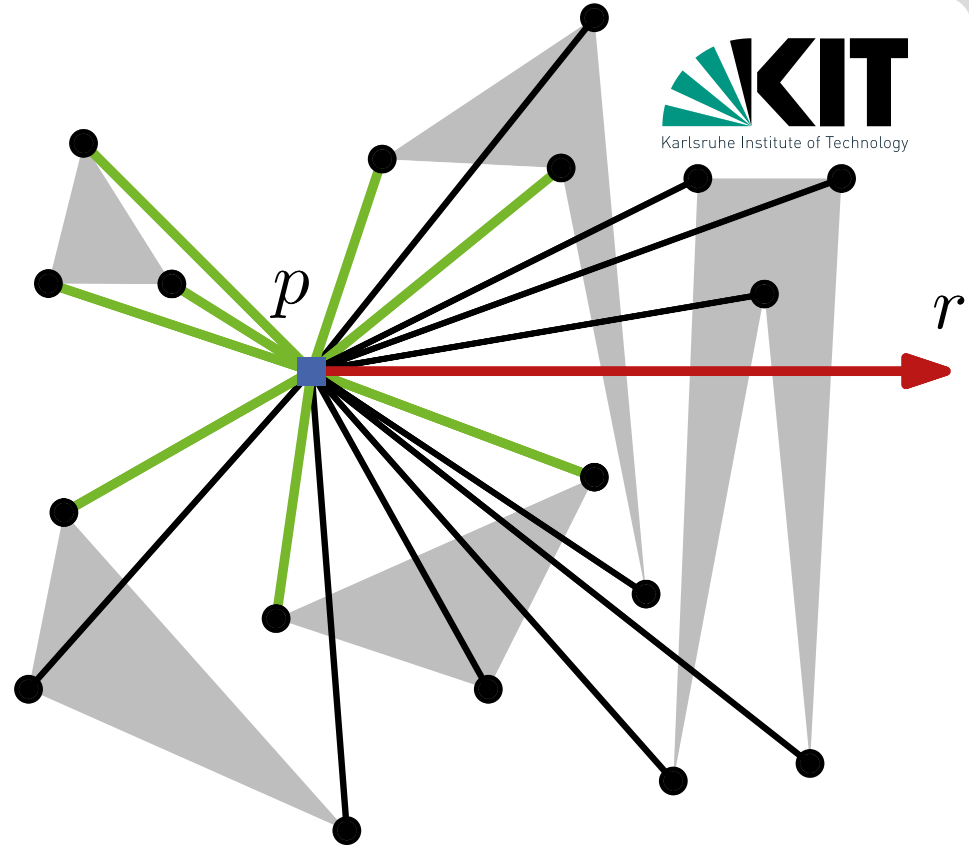
Problem: Given p and S, find in $O(n \log n)$ time all nodes that p sees in $V(S)$!



Computing Visible Nodes

VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

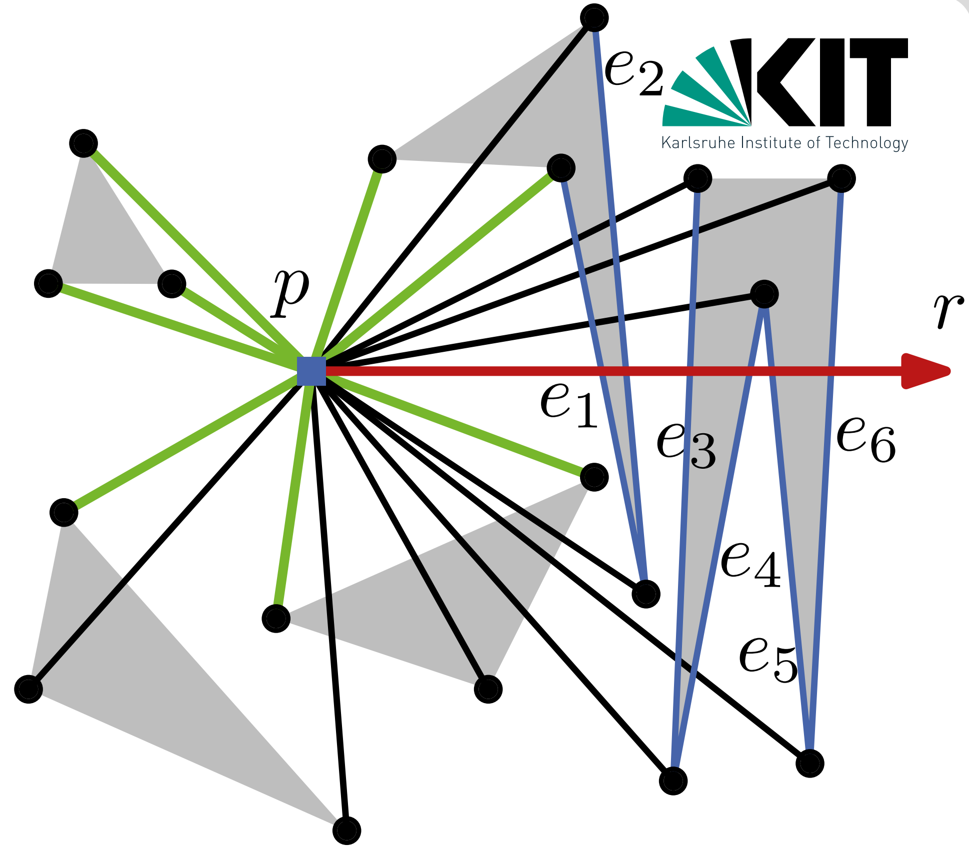


Computing Visible Nodes

VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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



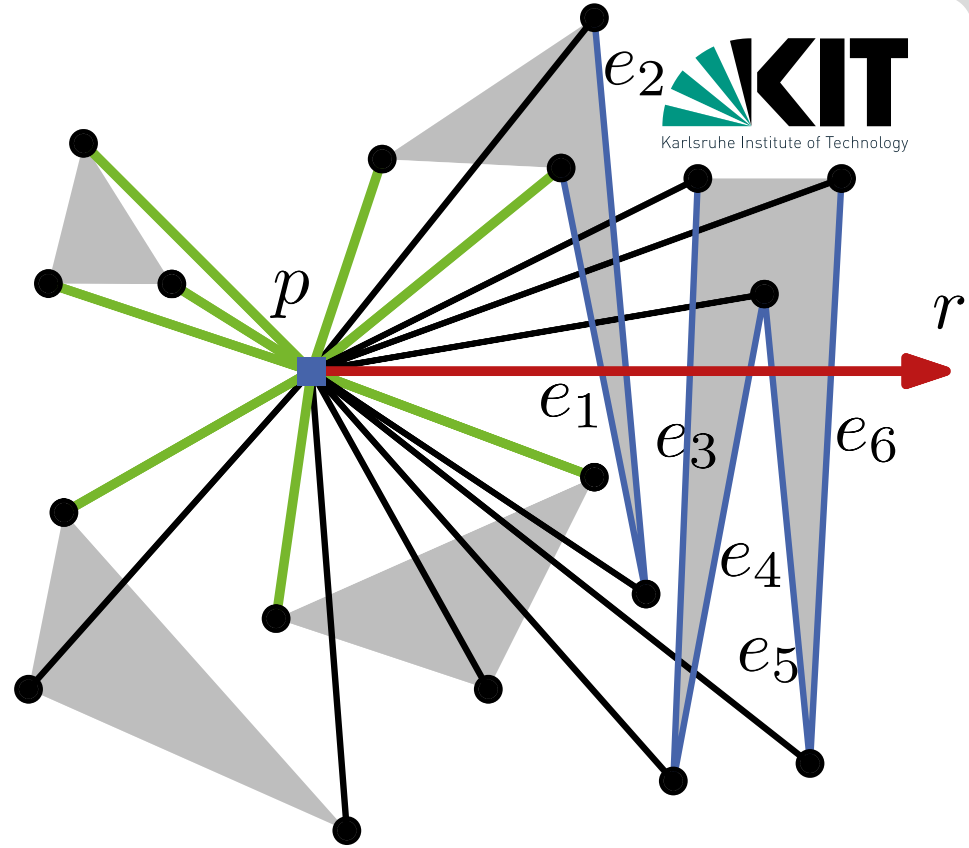
Computing Visible Nodes

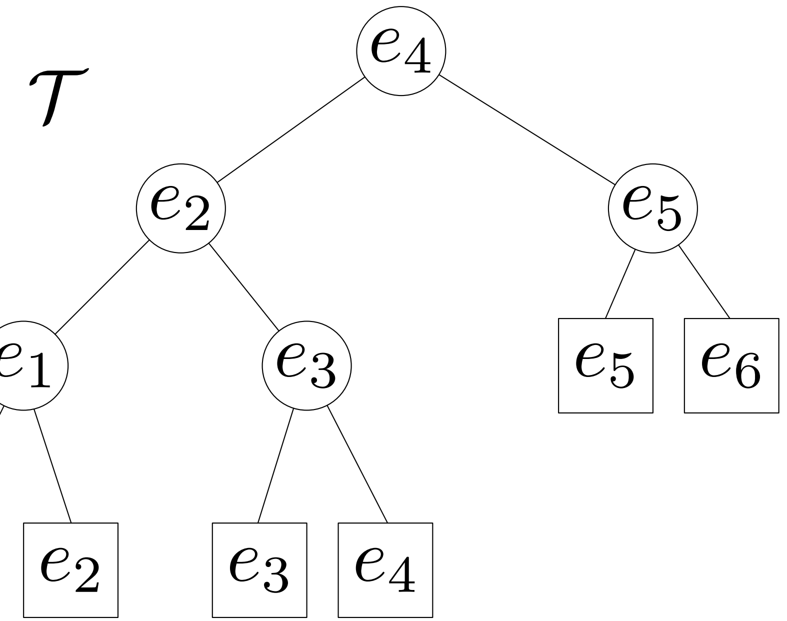
VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

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





Computing Visible Nodes

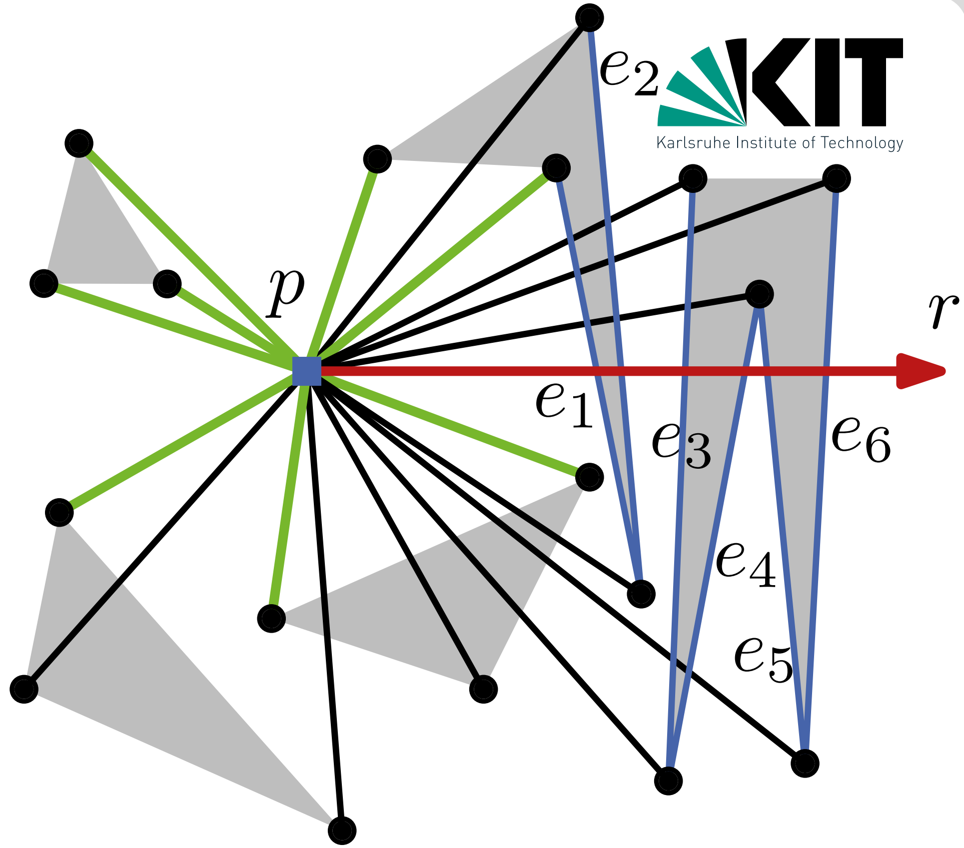
VisibleVertices(p, S)

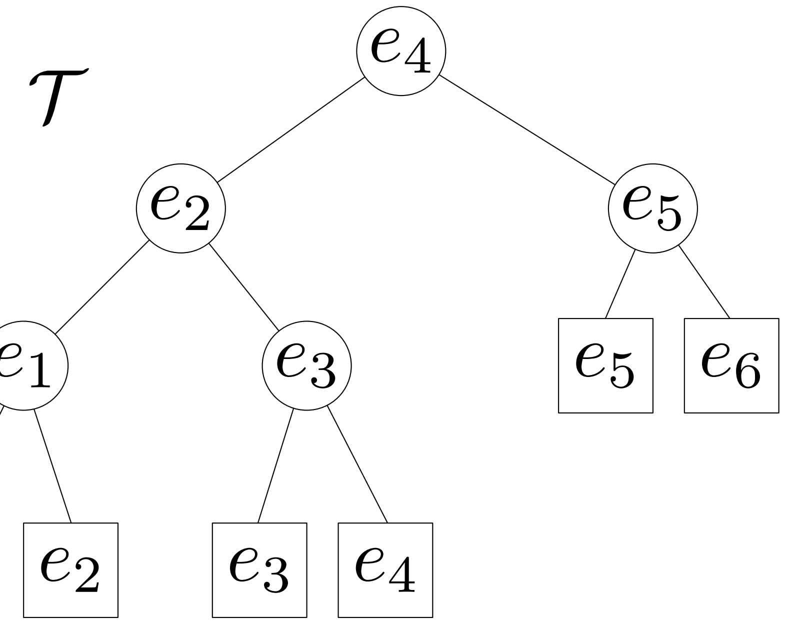
$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

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

$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p





Computing Visible Nodes

VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

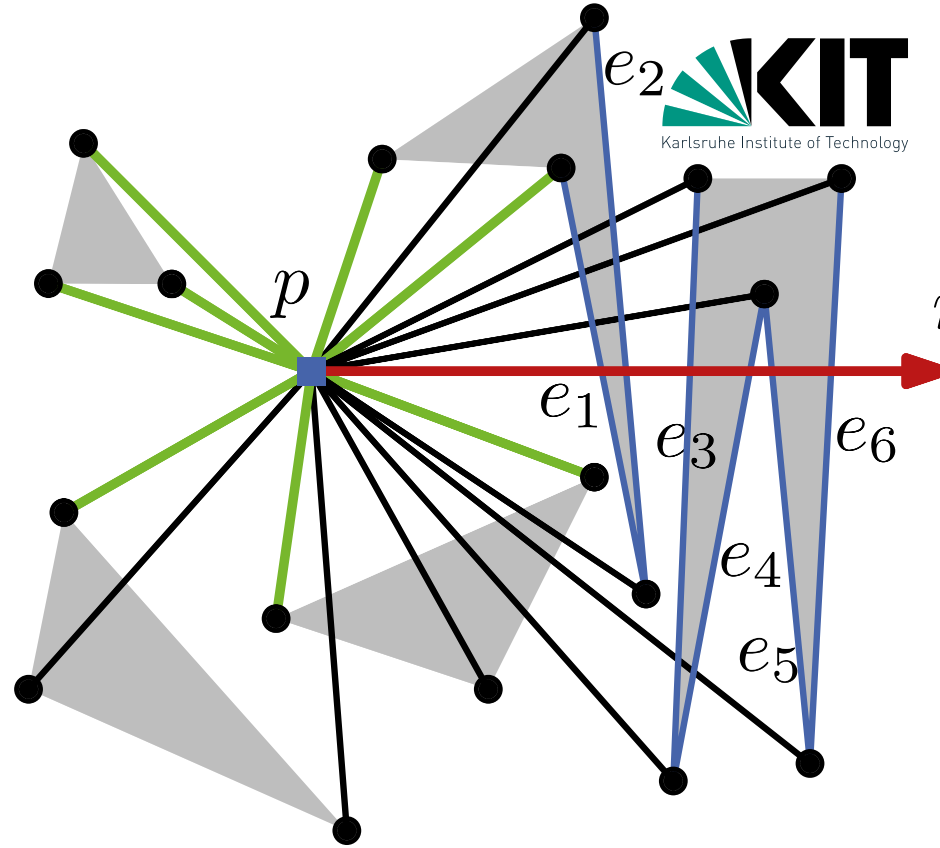
$I \leftarrow \{e \in E(S) \mid e \cap r \neq \emptyset\}$

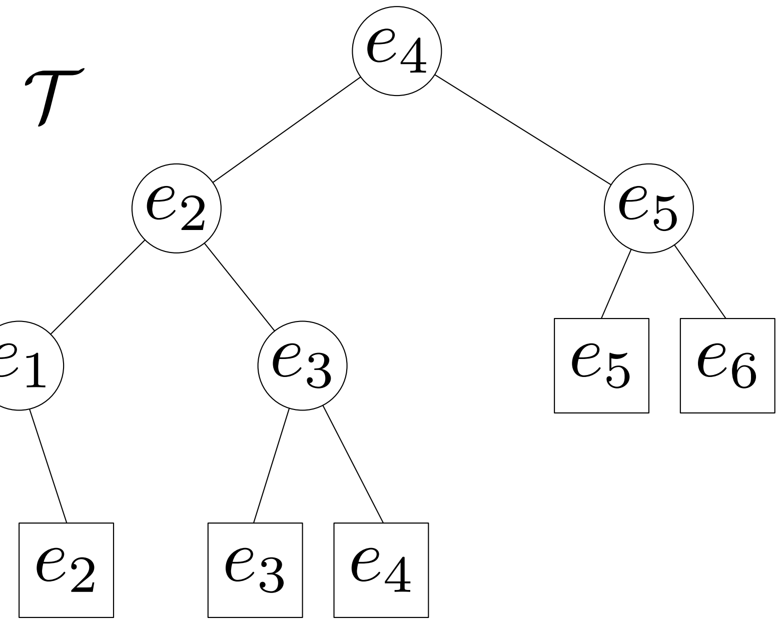
$\mathcal{T} \leftarrow$ balancedBinaryTree(I)

$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p

$v \prec v' :\Leftrightarrow$

$\angle v < \angle v'$ or
$(\angle v = \angle v'$ and $|pv| < |pv'|)$





Computing Visible Nodes

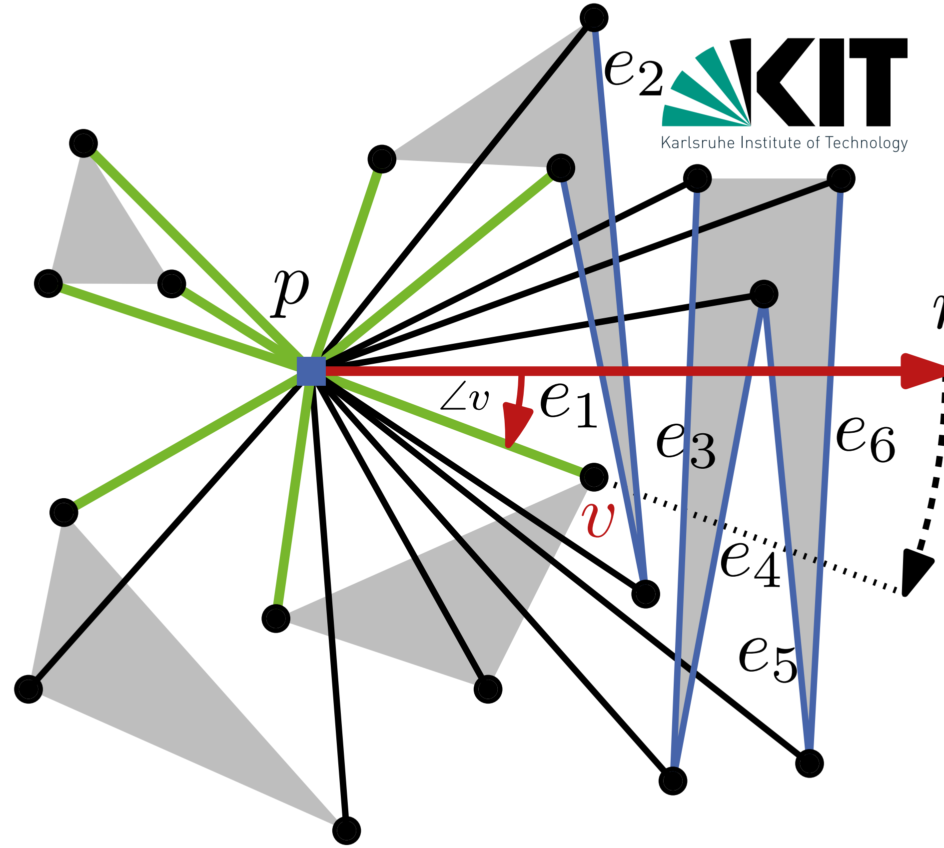
VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

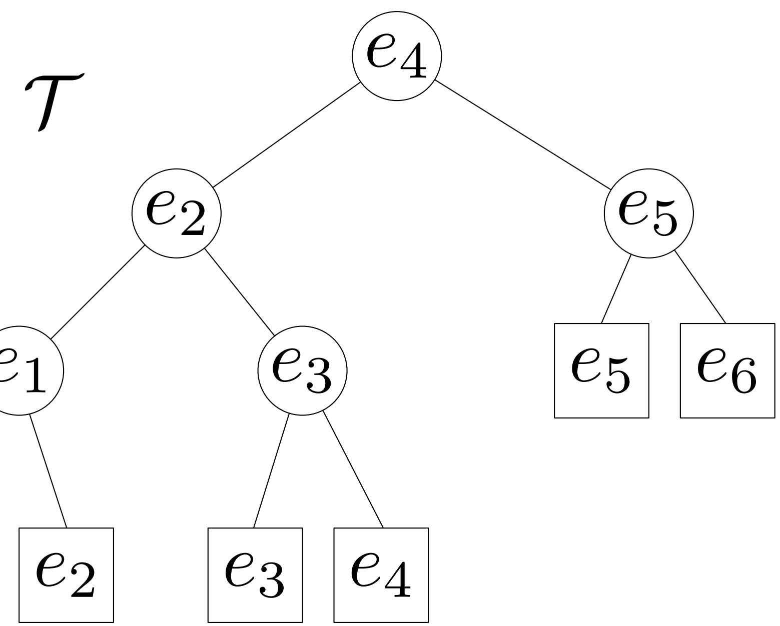
$\mathcal{T} \leftarrow$ balancedBinaryTree(I)

$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p



$v \prec v' :\Leftrightarrow$

$\angle v < \angle v'$ or
$(\angle v = \angle v'$ and $|pv| < |pv'|)$



Computing Visible Nodes

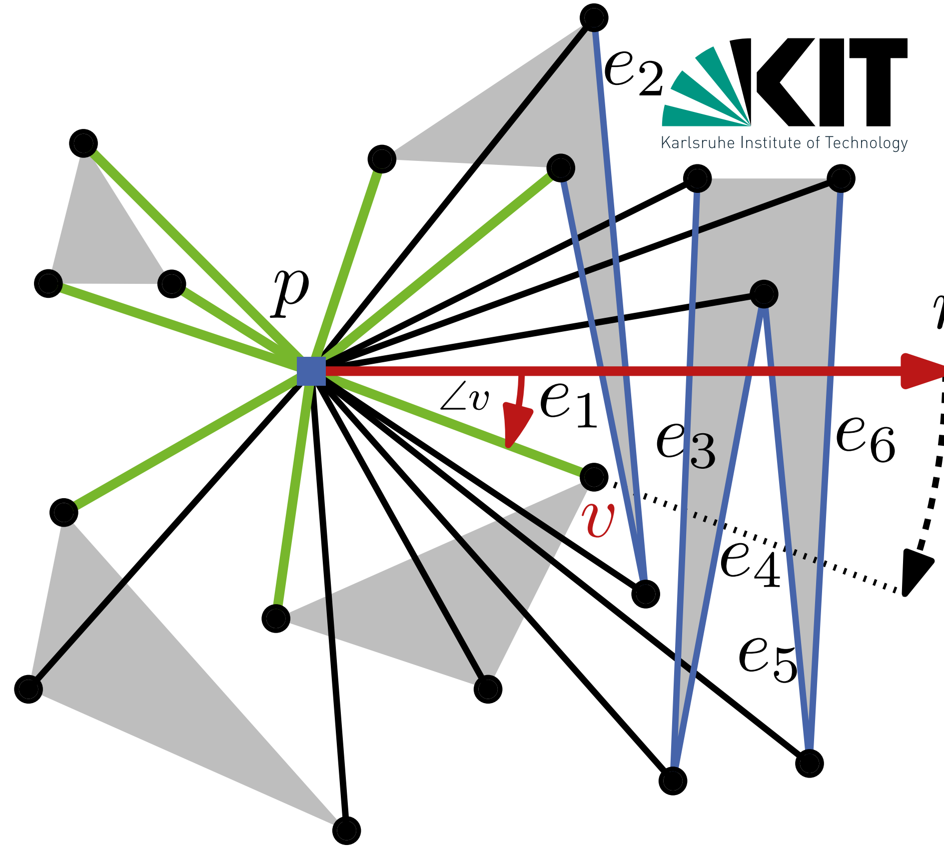
VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

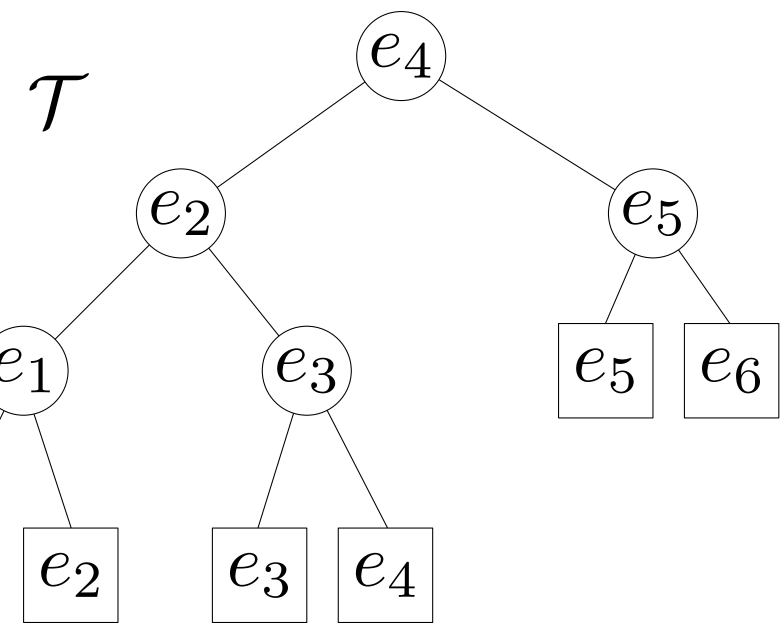
$\mathcal{T} \leftarrow$ balancedBinaryTree(I)

$w_1, \dots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p



$v \prec v' :\Leftrightarrow$

$\angle v < \angle v'$ or
$(\angle v = \angle v'$ and $|pv| < |pv'|)$



Sweep method with rotation

Computing Visible Nodes

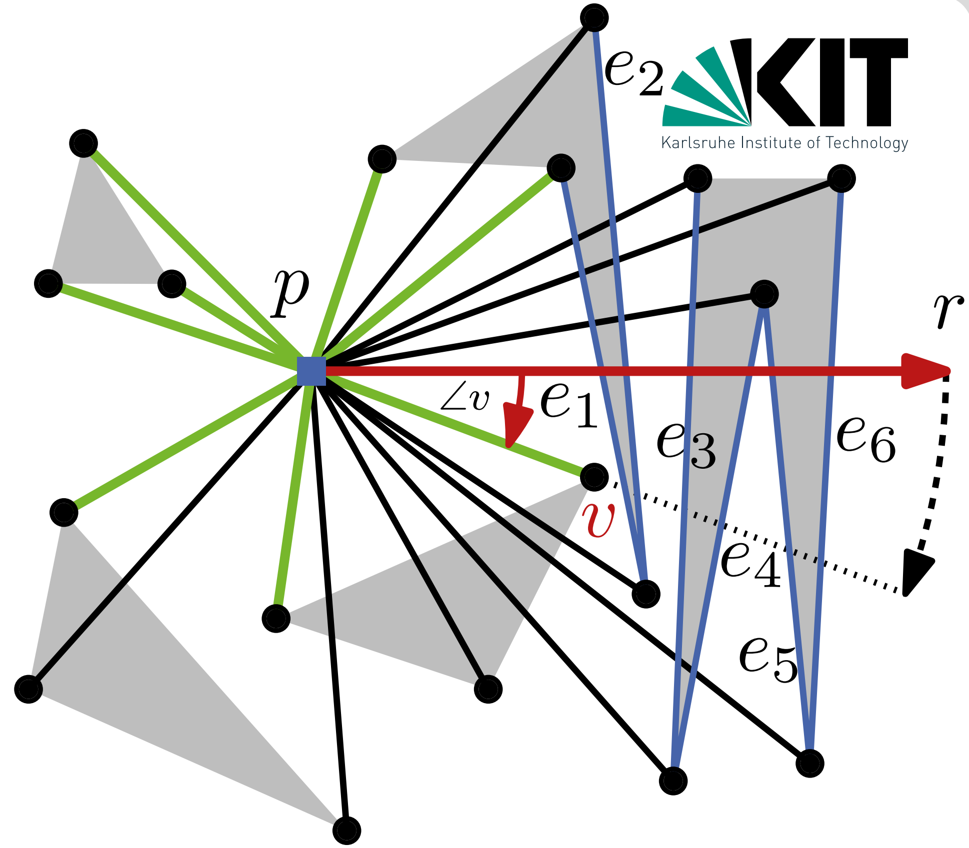
VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

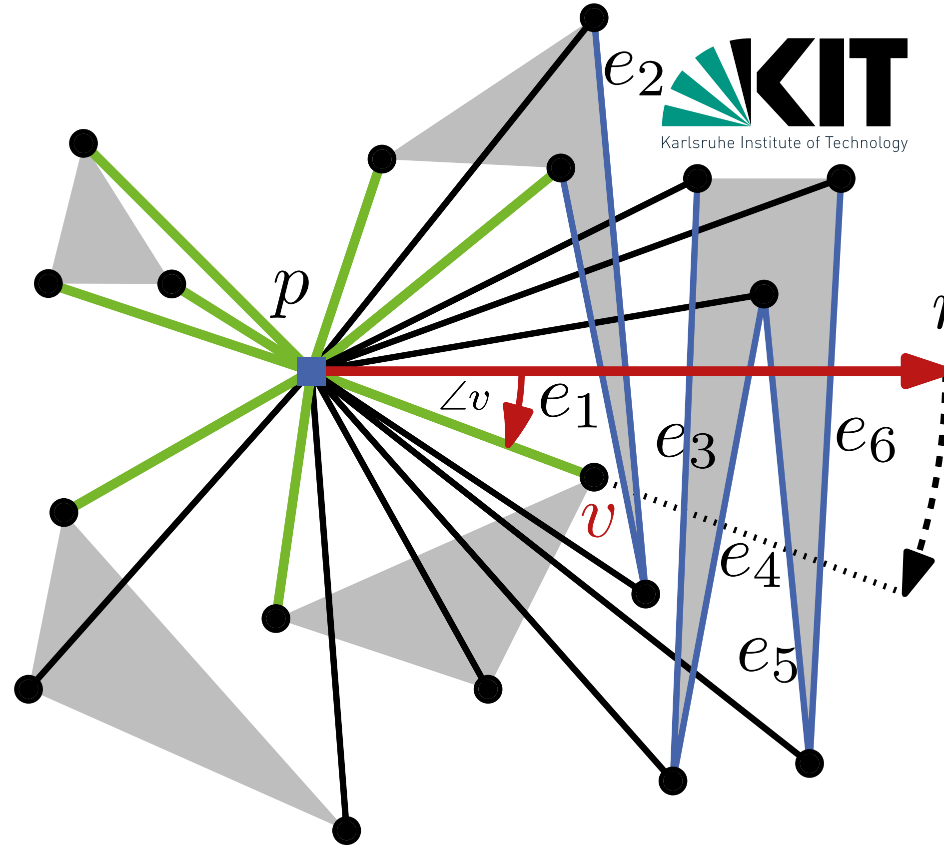
$I \leftarrow \{e \in E(S) \mid e \cap r \neq \emptyset\}$

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

$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p



Computing Visible Nodes



VisibleVertices(p, S)

- $r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$
- $I \leftarrow \{e \in E(S) \mid e \cap r \neq \emptyset\}$
- $\mathcal{T} \leftarrow$ balancedBinaryTree(I)
- $w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p
- $W \leftarrow \emptyset$
- **for** $i = 1$ **to** n **do**
  - **if** Visible(p, w_i) **then**
    - $W \leftarrow W \cup \{w_i\}$
  - Add to $\mathcal{T}$ edges incident to w_i: CW from $\overrightarrow{pw_i}^+$
  - Remove from $\mathcal{T}$ edges incident to w_i:CCW from $\overrightarrow{pw_i}^-$
- **return** W

Computing Visible Nodes

VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

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

$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p

$W \leftarrow \emptyset$

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

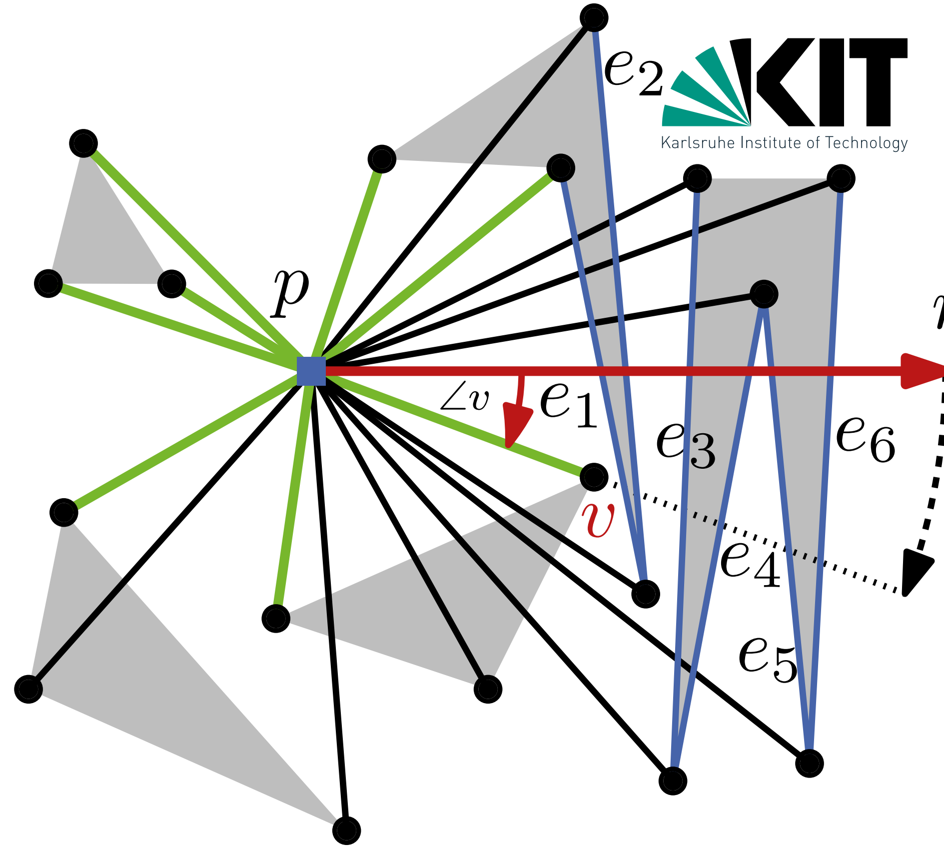
  if Visible(p, w_i) **then**

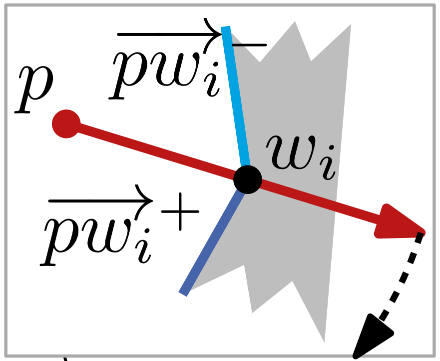
    $W \leftarrow W \cup \{w_i\}$

  Add to $\mathcal{T}$ edges incident to w_i: CW from $\overrightarrow{pw_i}^+$

  Remove from $\mathcal{T}$ edges incident to w_i:CCW from $\overrightarrow{pw_i}^-$

return W





Computing Visible Nodes

VisibleVertices(p, S)

$r \leftarrow \{p + (k, 0) \mid k \in \mathbb{R}_0^+\}$

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

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

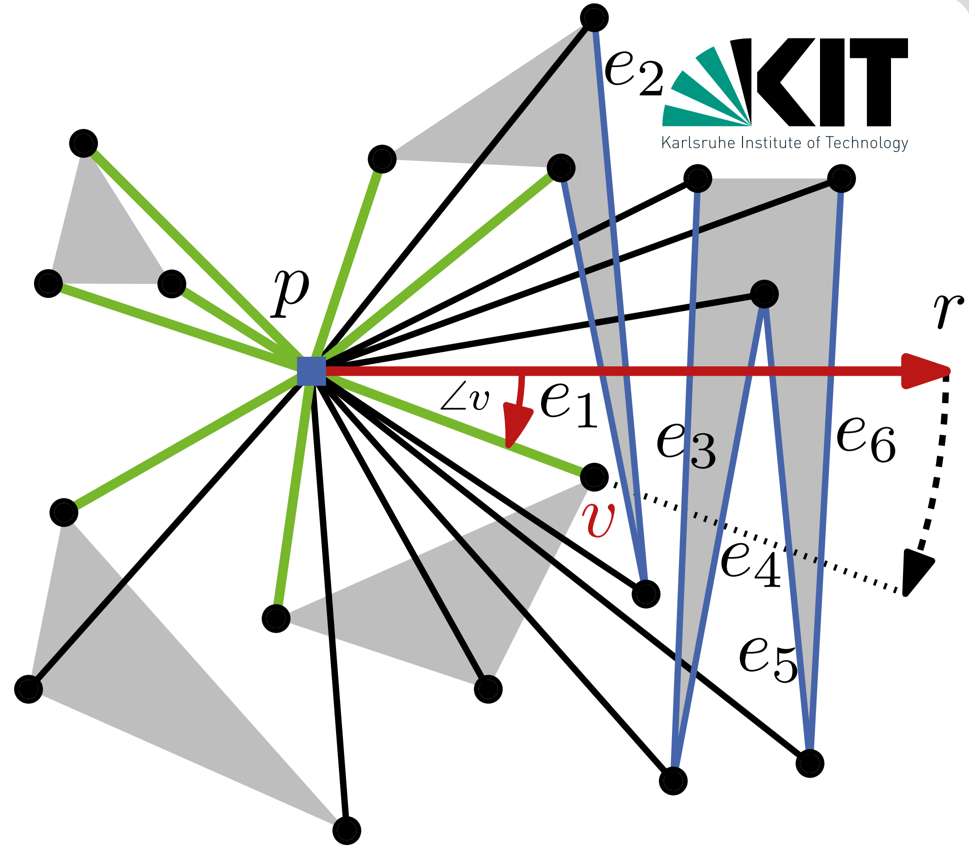
$w_1, \ldots, w_n \leftarrow$ sort $V(S)$ in cyclic order around p

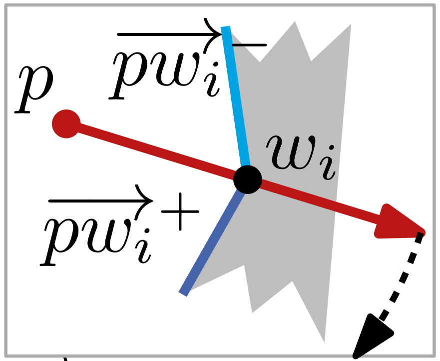
$W \leftarrow \emptyset$

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

- **if** Visible(p, w_i) **then**
  - $W \leftarrow W \cup \{w_i\}$
- Add to $\mathcal{T}$ edges incident to w_i: CW from $\overrightarrow{pw_i}^+$
- Remove from $\mathcal{T}$ edges incident to w_i:CCW from $\overrightarrow{pw_i}^-$

return W

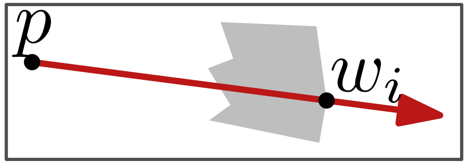




Visibility Case Analysis

Visible(p, w_i)

if $\overline{pw_i}$ intersects polygon of w_i **then**
 ⌊ **return** false



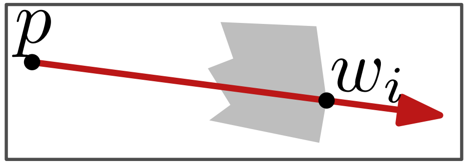
nil

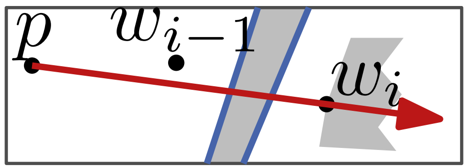
Visibility Case Analysis

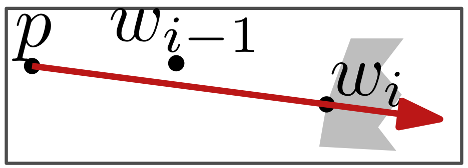
Visible(p, w_i)

if $\overline{pw_i}$ intersects polygon of w_i **then**
    return false

if $i = 1$ or $w_{i-1} \notin \overline{pw_i}$ **then**
    $e \leftarrow$ edge of leftmost leaf of $\mathcal{T}$
    if $e \neq$ nil and $\overline{pw_i} \cap e \neq \emptyset$ **then**
        return false
    else return true





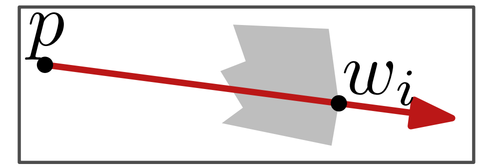


nil

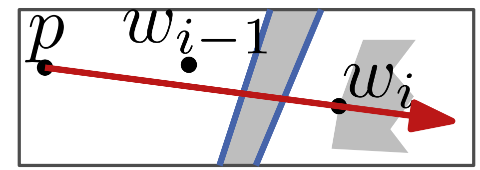
Visibility Case Analysis

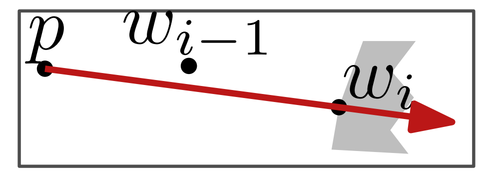
Visible(p, w_i)

if $\overline{pw_i}$ intersects polygon of w_i **then**
  return false

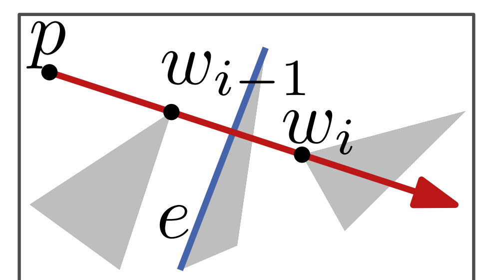


if $i = 1$ or $w_{i-1} \notin \overline{pw_i}$ **then**
  $e \leftarrow$ edge of leftmost leaf of $\mathcal{T}$
  if $e \neq$ nil and $\overline{pw_i} \cap e \neq \emptyset$ **then**
    return false
  else return true





else
  if w_{i-1} is not visible **then**
    return false
  else
    $e \leftarrow$ find edge in $\mathcal{T}$, that $\overline{w_{i-1}w_i}$ cuts; **if** $e \neq$ nil
     then return false
    else return true



Summary

Thm 1: A shortest st-path in an area with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Summary

Thm 1: A shortest st-path in an area with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Proof:

- Correctness follows directly from Lemma 1.

Summary

Thm 1: A shortest st-path in an area with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Proof:

- Correctness follows directly from Lemma 1.
- Running time:
  – VisibleVertices takes $O(n \log n)$ time per vertex – n calls to VisibleVertices

Summary

Thm 1: A shortest st-path in an area with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Proof:

- Correctness follows directly from Lemma 1.
- Running time:
  – VisibleVertices takes $O(n \log n)$ time per vertex – n calls to VisibleVertices $\square$

Discussion

Robots are not single points...

Discussion

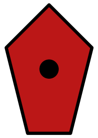
Robots are not single points...

For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).

Discussion

Robots are not single points...

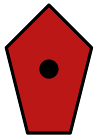
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

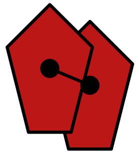
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

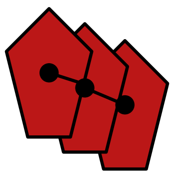
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

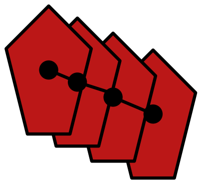
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles (→ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

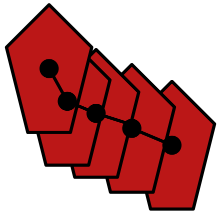
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles
($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles
($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).

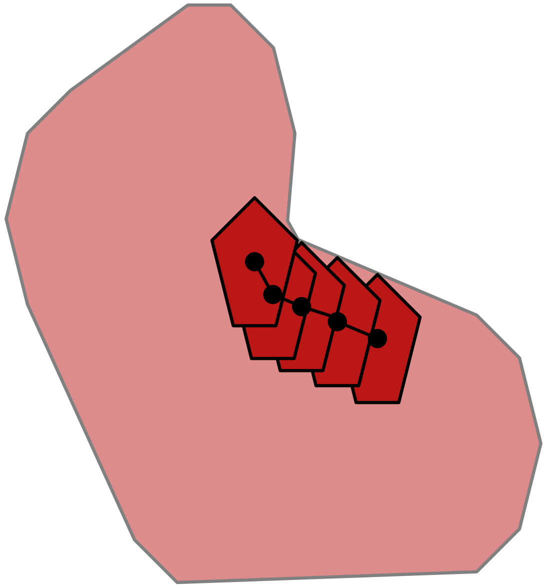


Discussion

Karlsruhe Institute of Technology

Robots are not single points...

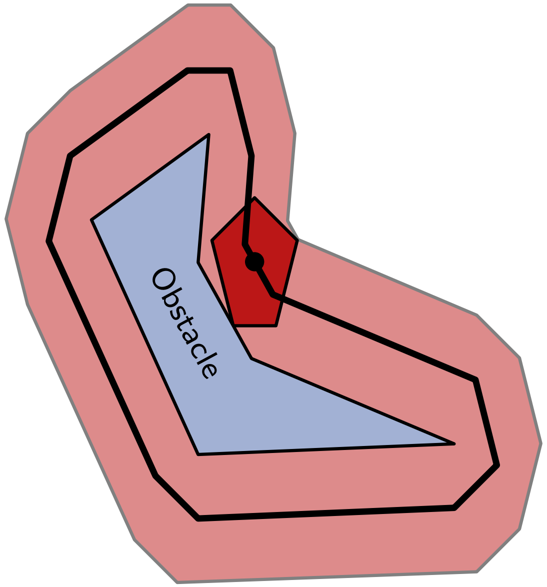
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles
($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

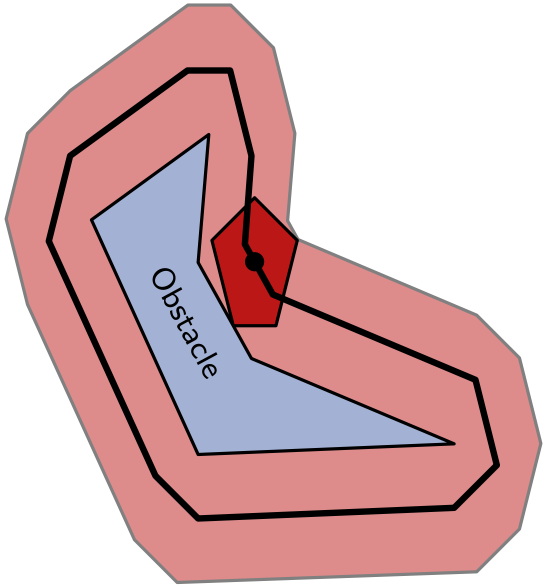
For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles (→ Minkowski Sums, Ch. 13 in [BCKO08]).



Discussion

Robots are not single points...

For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles (→ Minkowski Sums, Ch. 13 in [BCKO08]).

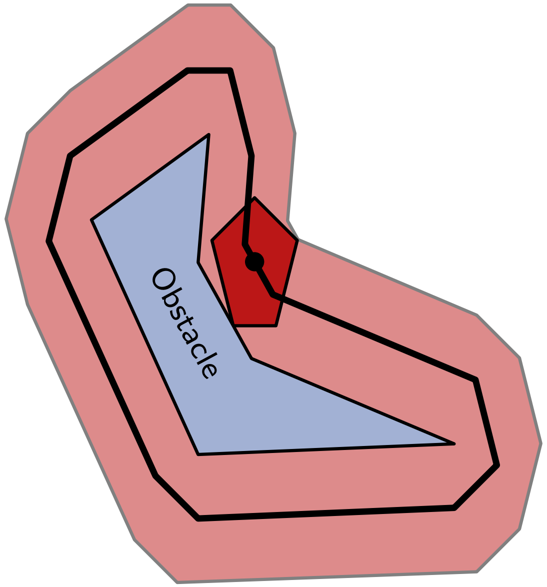


Can we compute faster than $O(n^2 \log n)$?

Discussion

Robots are not single points...

For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Can we compute faster than $O(n^2 \log n)$?

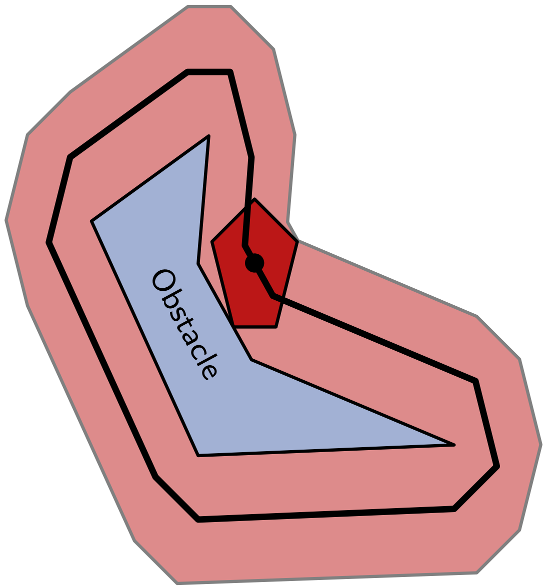
Yes, by use duality and a simultaneous rotation sweep for all points in the dual. Computing the arrangement, is also in $O(n^2)$. Even though G_{vis} can have $\Omega(n^2)$ edges, the visibility graph can be constructed even faster with an output sensitive $O(n \log n + m)$-time algorithm.

[Ghosh, Mount 1987]

Discussion

Robots are not single points...

For robots modelled by a convex polygon that cannot rotate, we can resize (grow) the polygons representing the obstacles ($\rightarrow$ Minkowski Sums, Ch. 13 in [BCKO08]).



Can we compute faster than $O(n^2 \log n)$?

Yes, by use duality and a simultaneous rotation sweep for all points in the dual. Computing the arrangement, is also in $O(n^2)$. Even though G_{vis} can have $\Omega(n^2)$ edges, the visibility graph can be constructed even faster with an output sensitive $O(n \log n + m)$-time algorithm.

[Ghosh, Mount 1987]

If you search only for *one* shortest Euclidean st-path, there is an algorithm with optimal $O(n \log n)$ time. [Hershberger, Suri 1999]