Robot Motion Planning

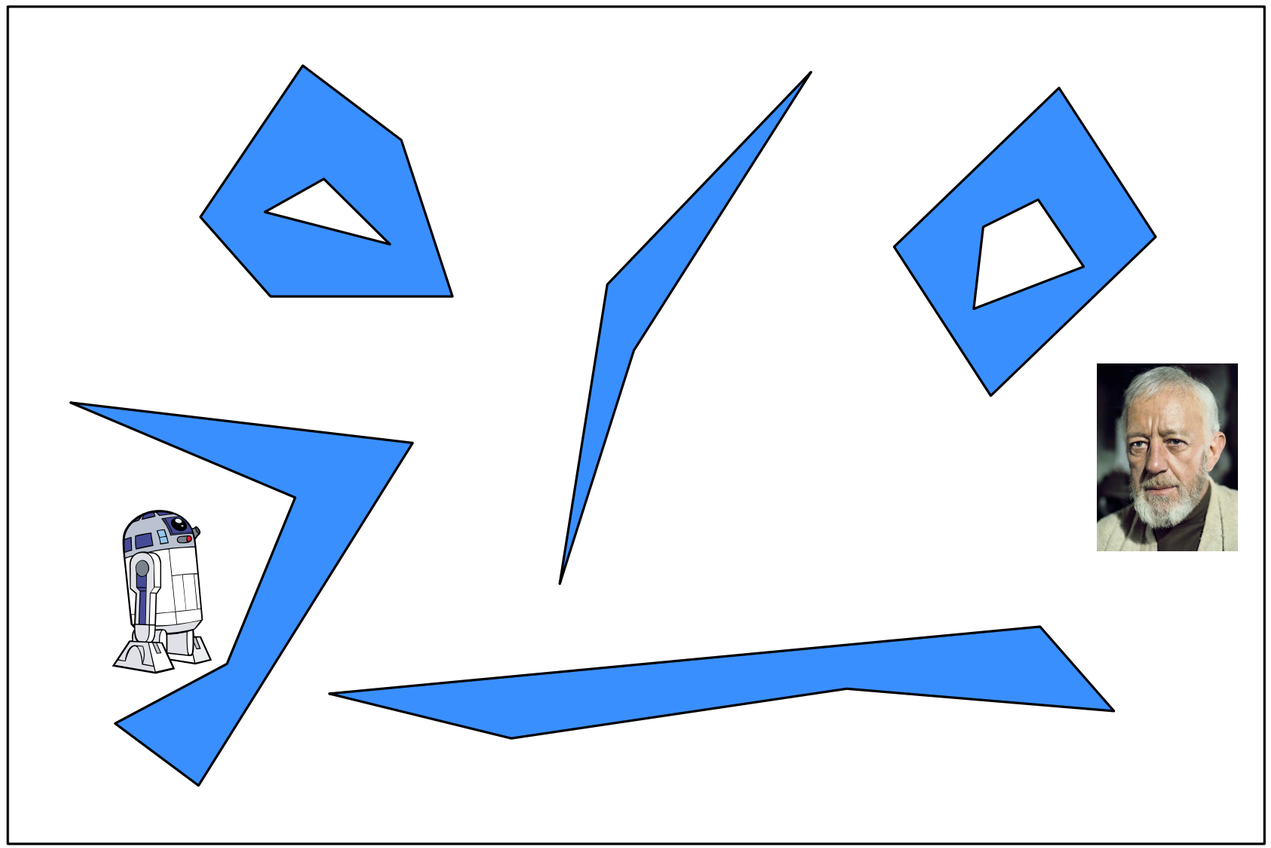
road maps

configuration spaces

Minkowski sums

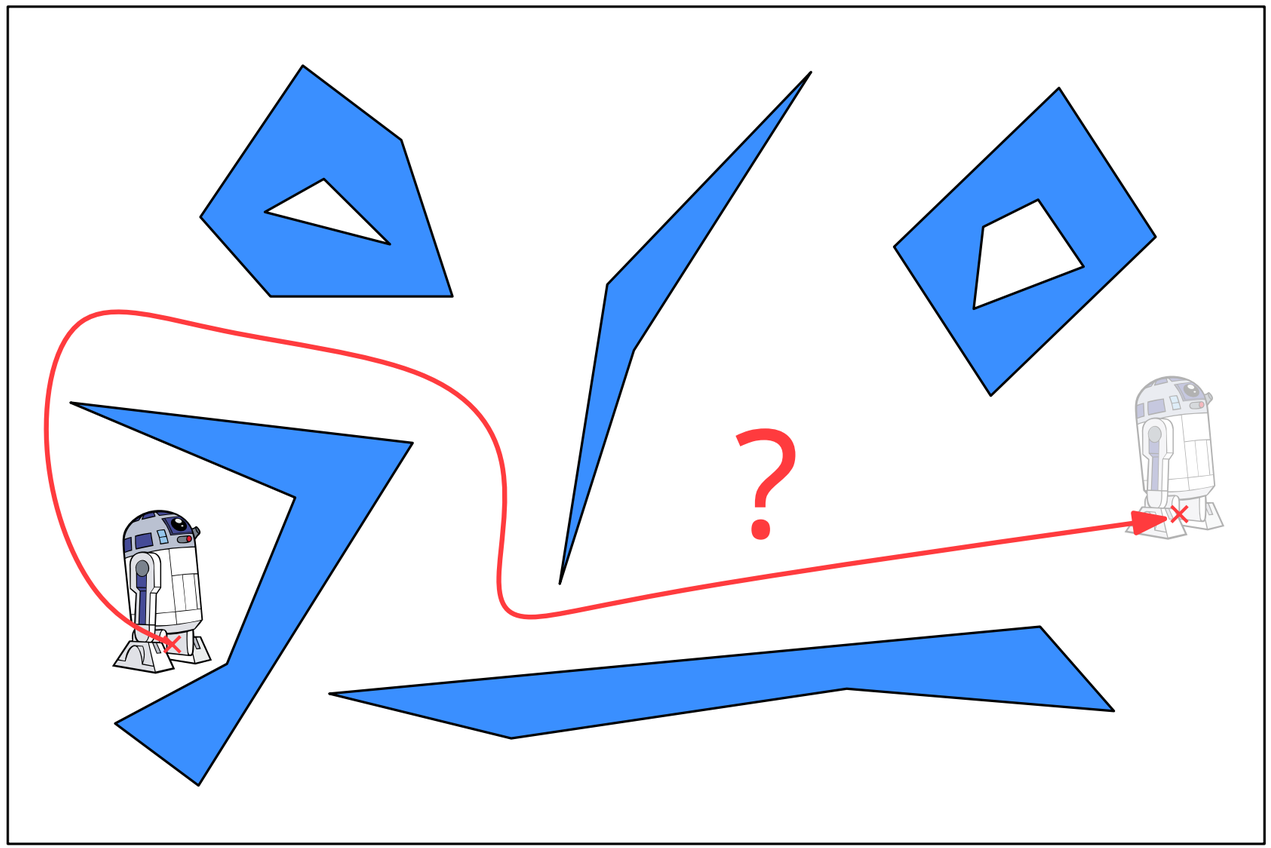
visibility graph

Motion planning for robots



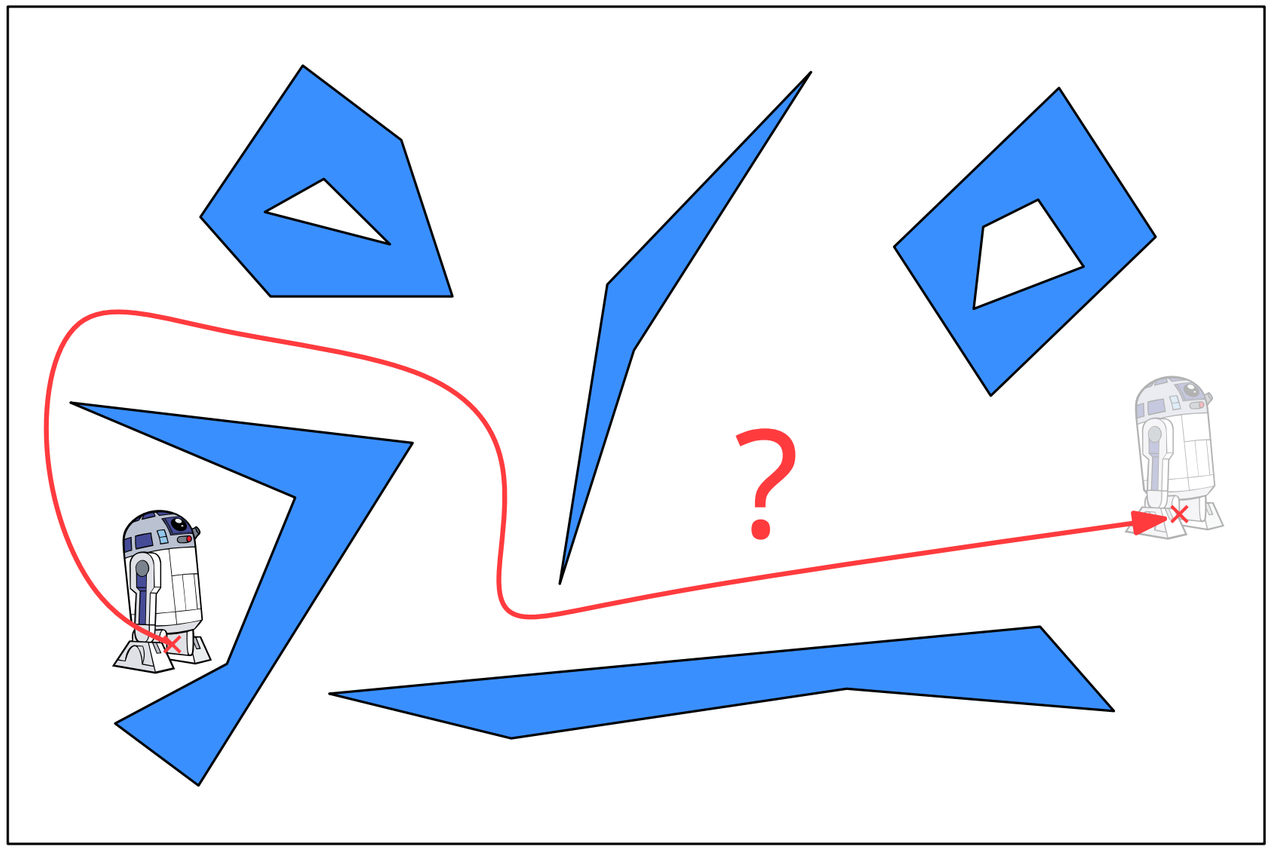
Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Motion planning for robots



Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Motion planning for robots

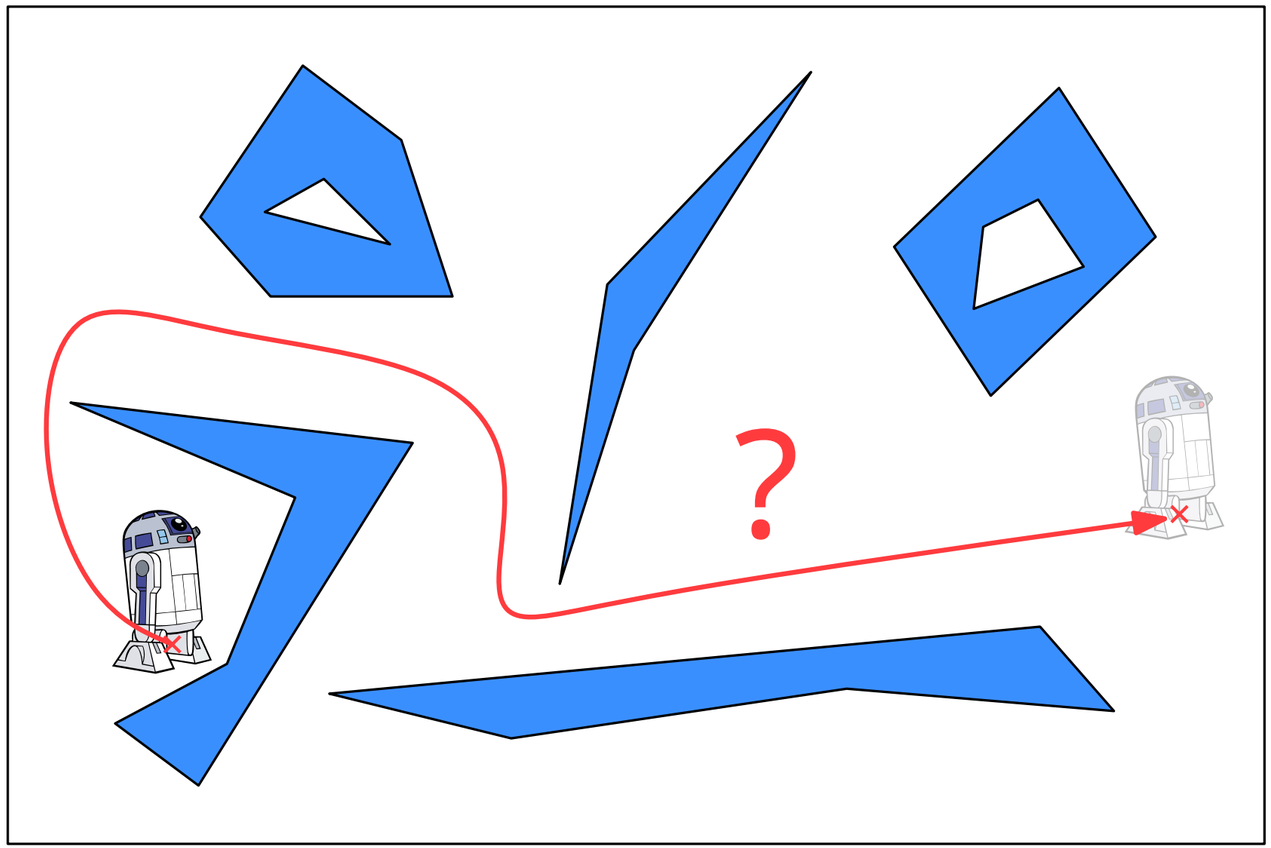


variants

- shape of robot
- shape of obstacles
- motion (e.g. rotation) of robot
- motion of obstacles
- 2D or 3D

Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Motion planning for robots

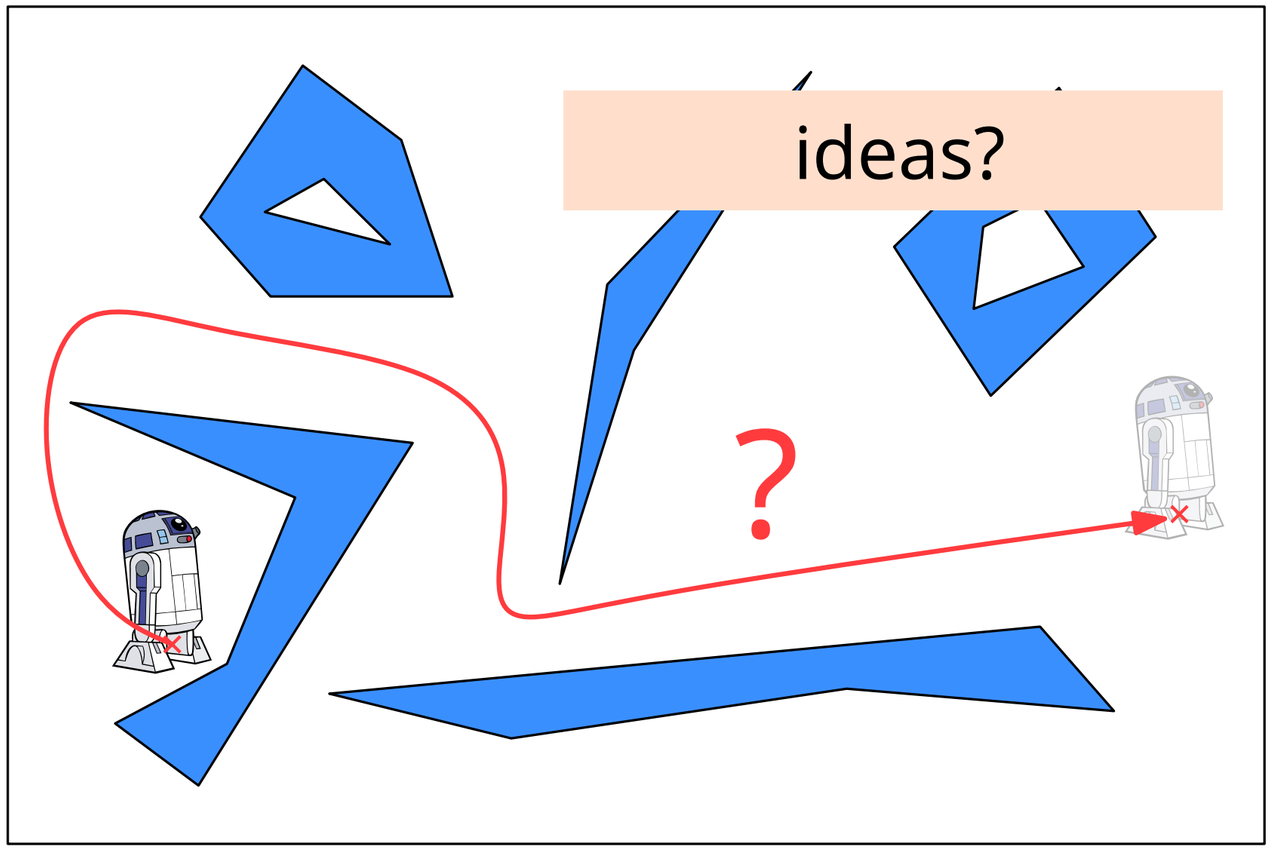


simplest variants

- shape of robot **point-shaped**
- shape of obstacles **polygonal**
- motion (e.g. rotation) of robot **translations**
- motion of obstacles **no motion**
- 2D or 3D **2D**

Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Motion planning for robots

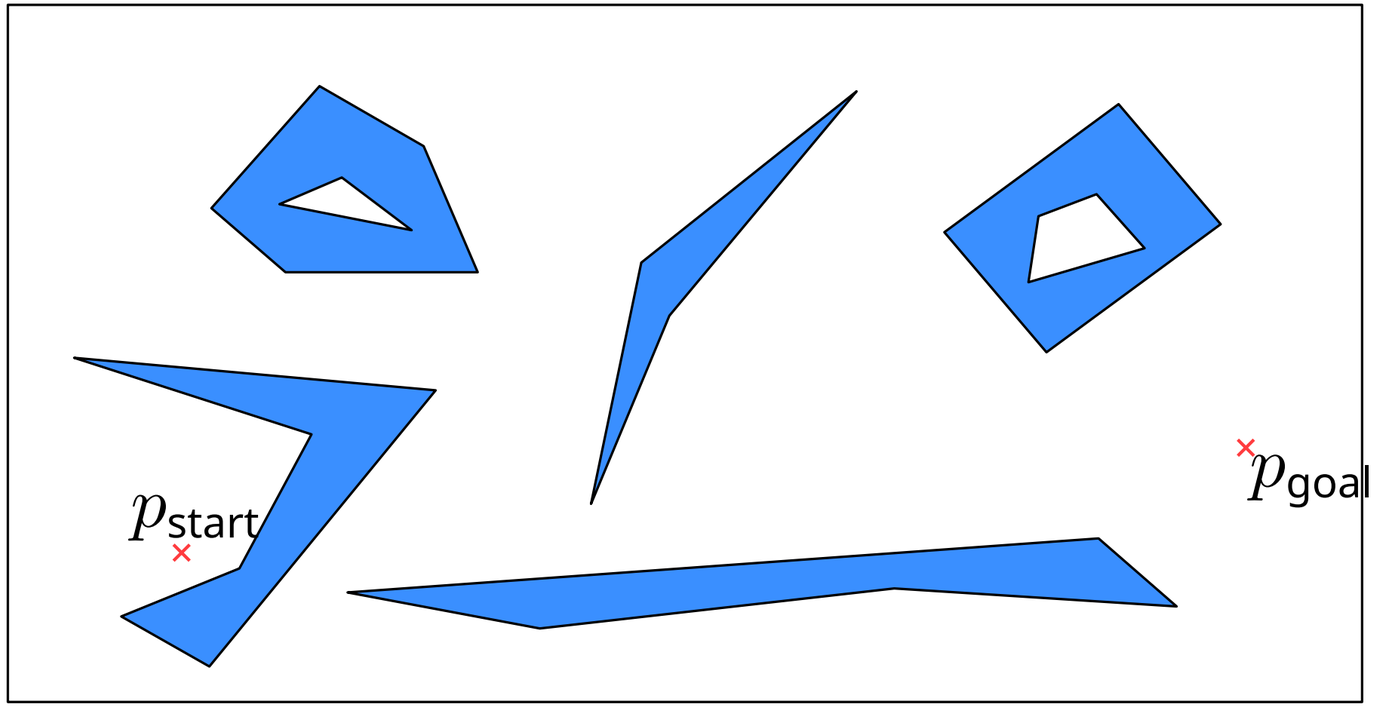


simplest variants

- shape of robot **point-shaped**
- shape of obstacles **polygonal**
- motion (e.g. rotation) of robot **translations**
- motion of obstacles **no motion**
- 2D or 3D **2D**

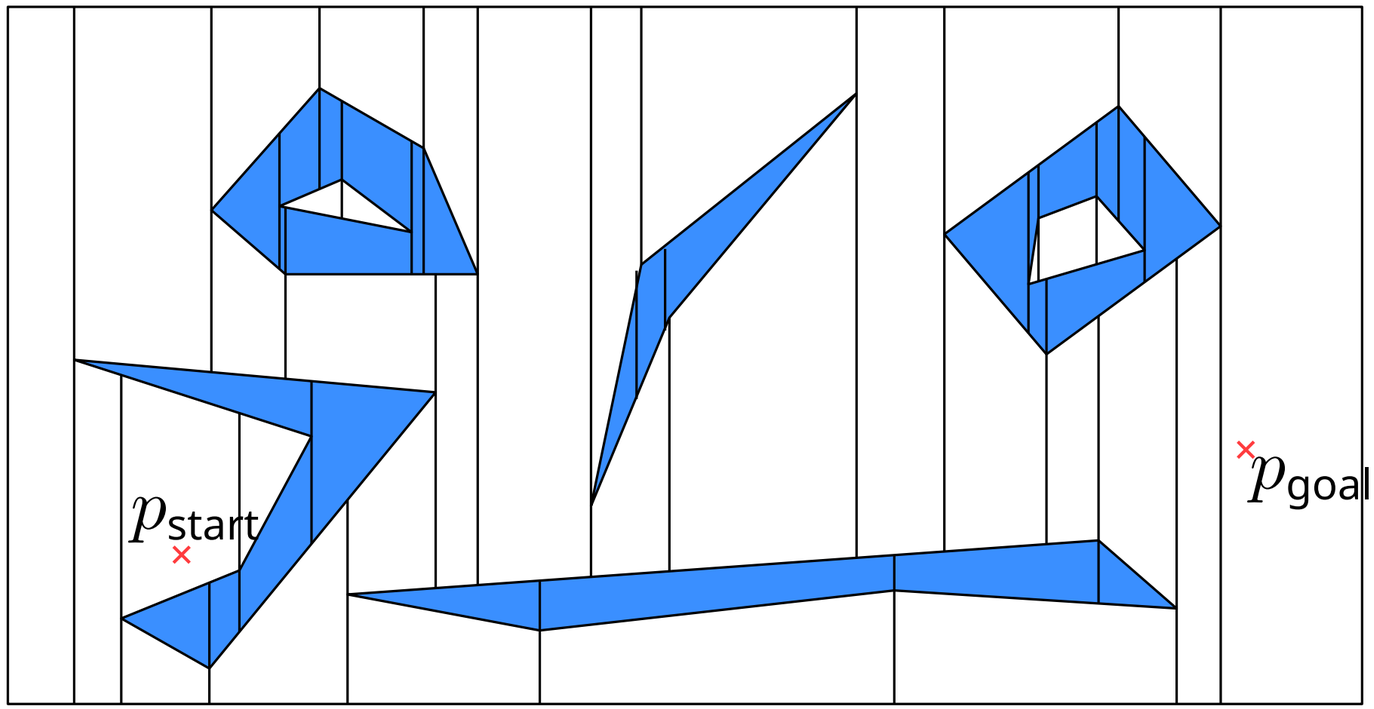
Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Idea: shortest paths in graphs



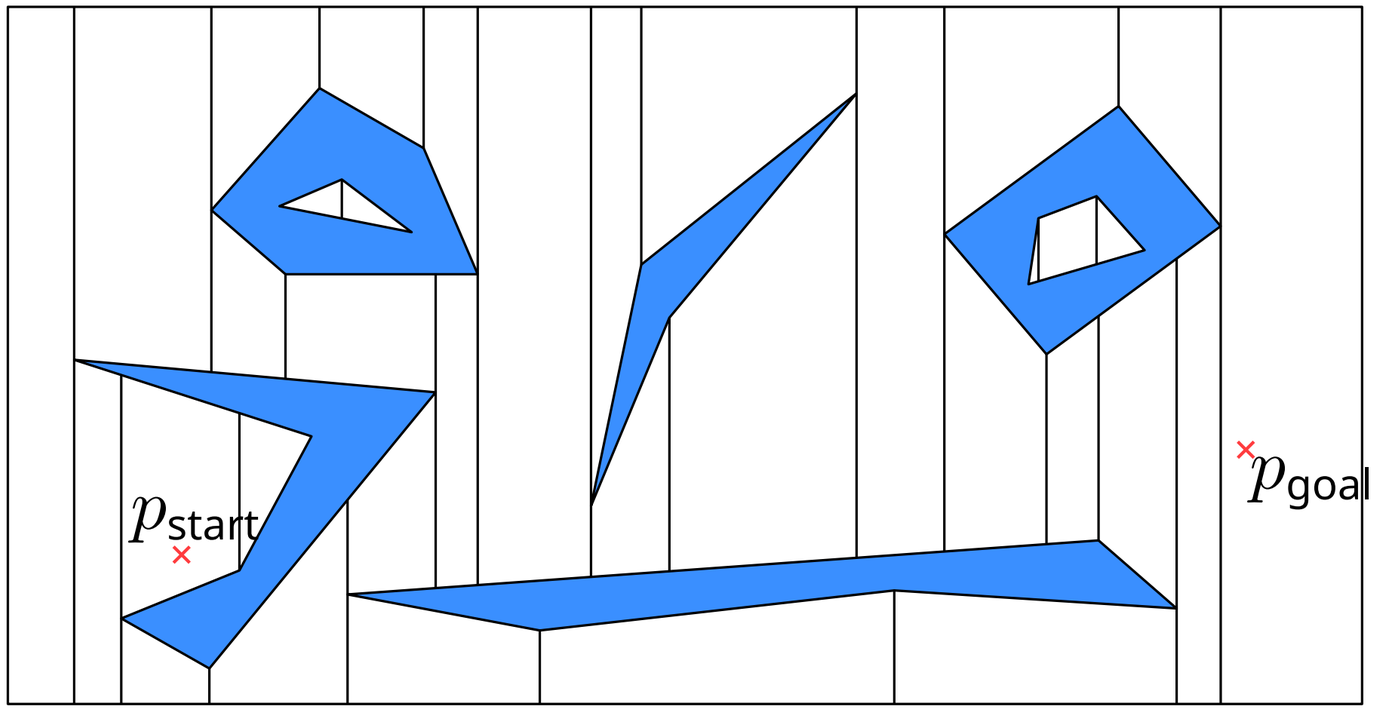
Idea: shortest paths in graphs

- compute vertical decomposition



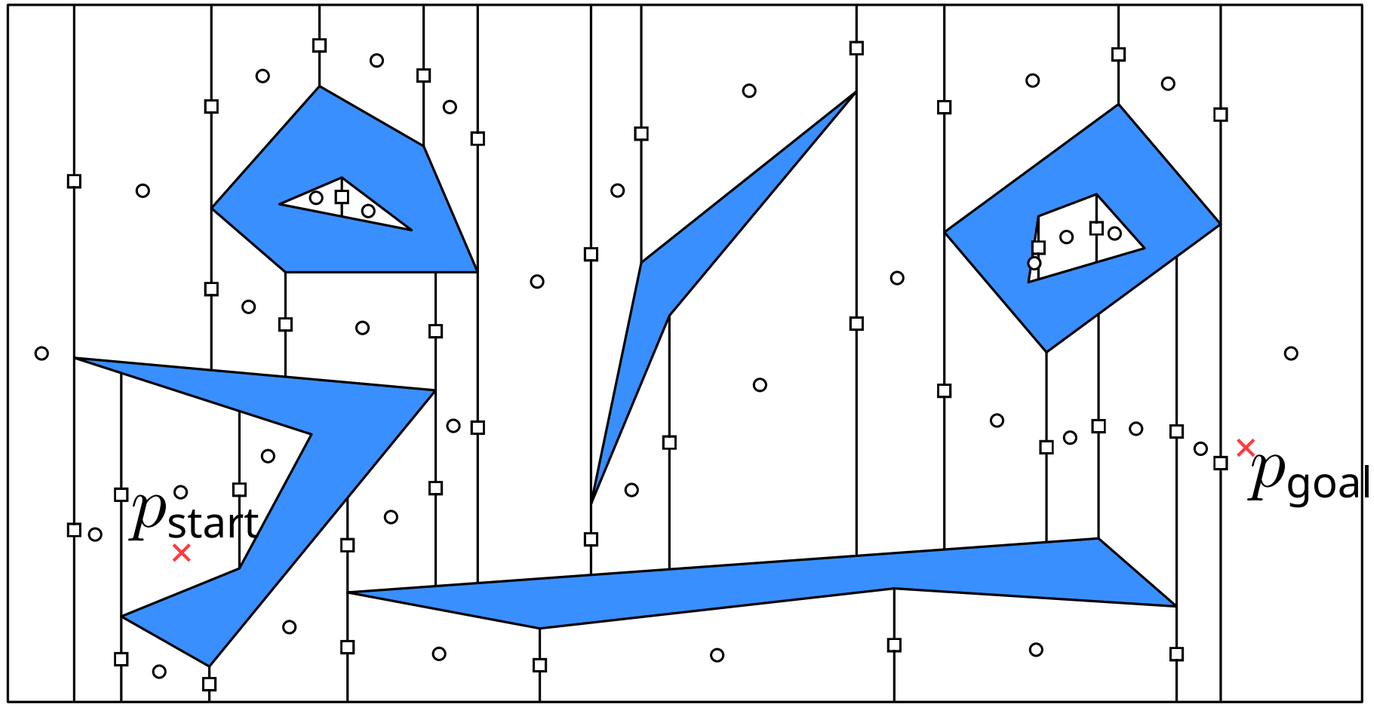
Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles



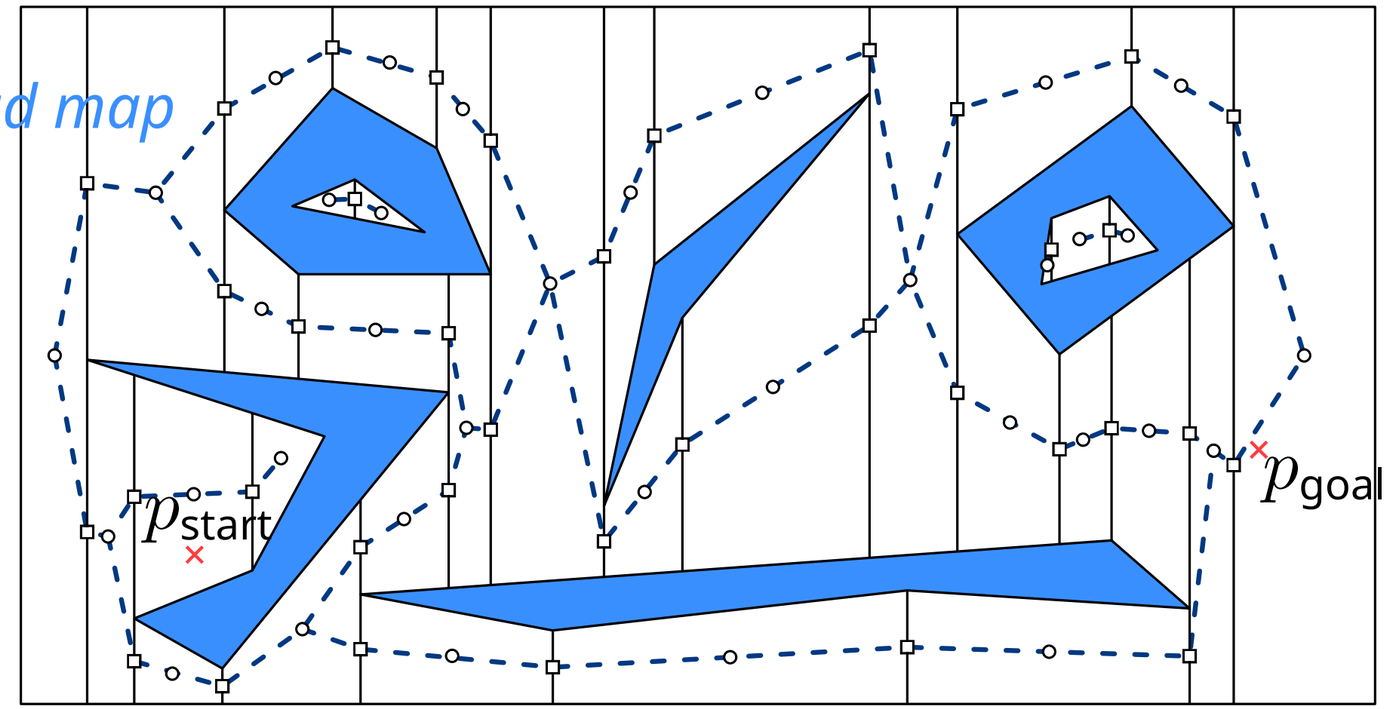
Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments



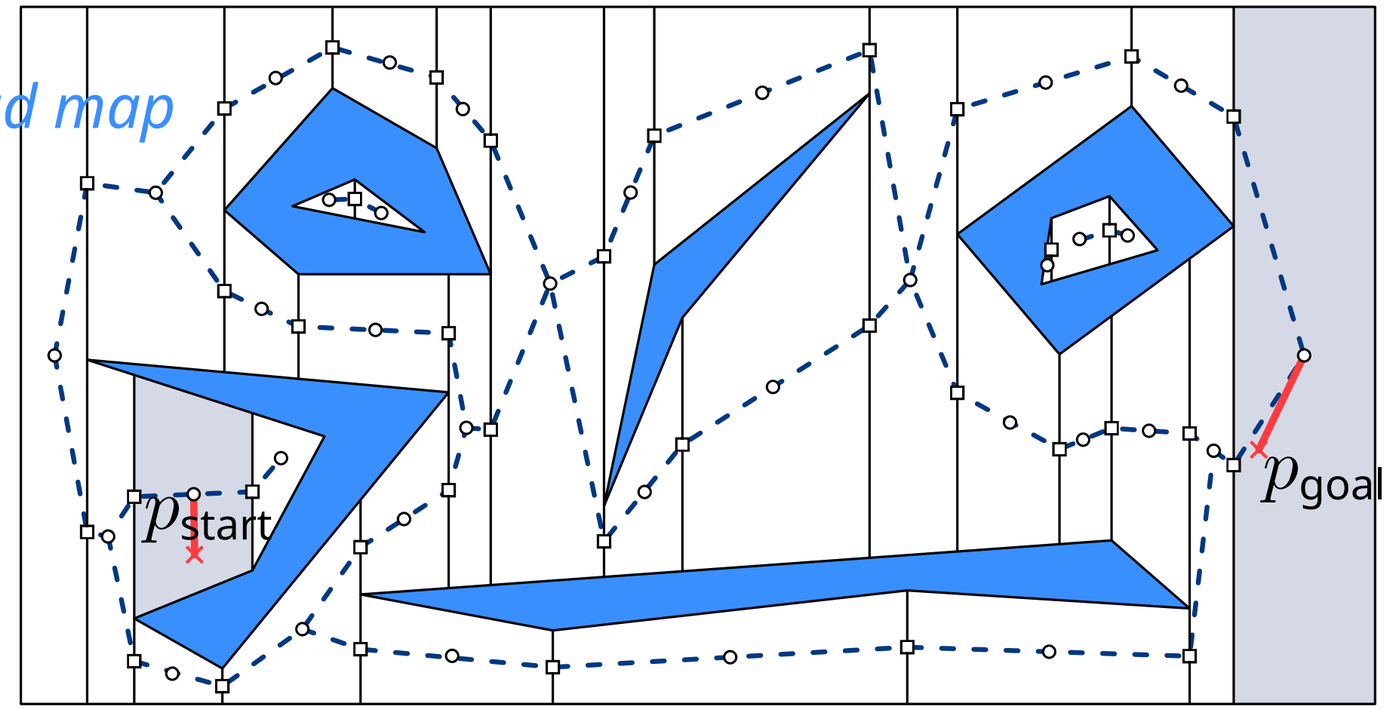
Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments



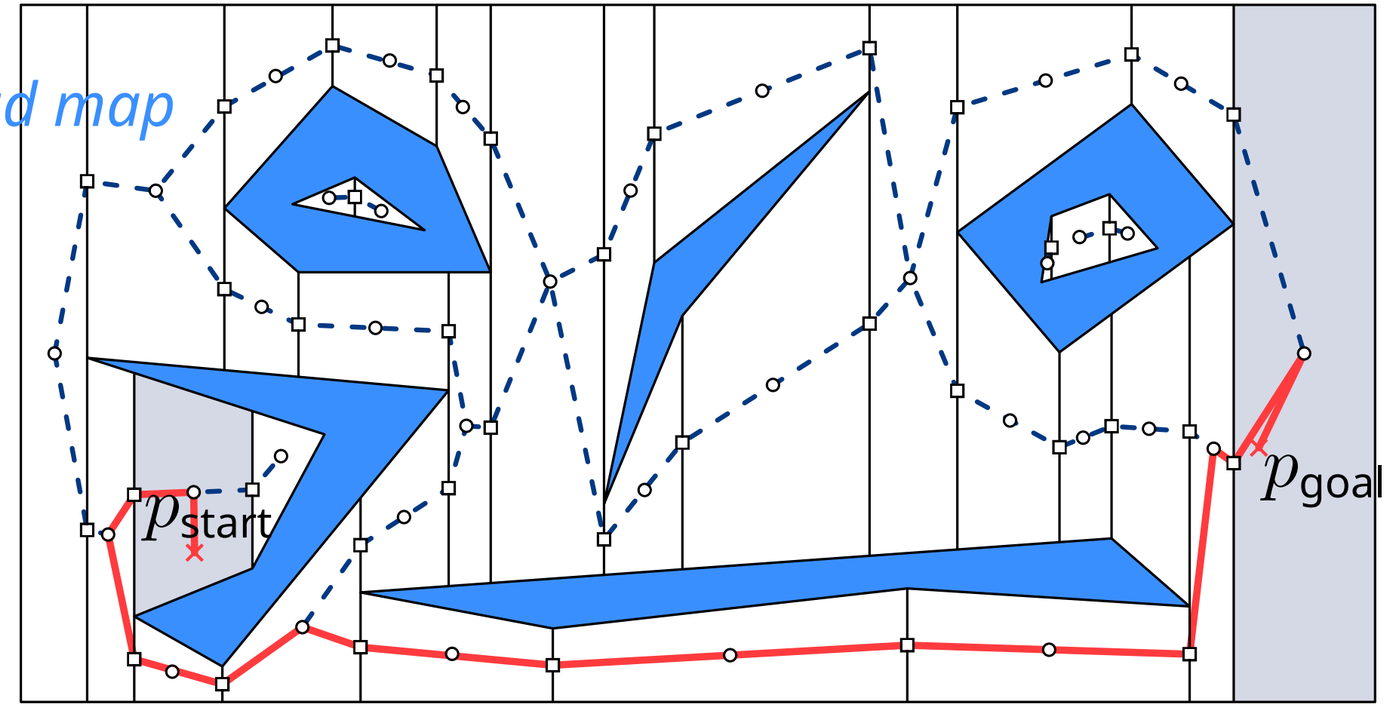
Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments
- locate start and goal



Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments
- locate start and goal
- find shortest path in G

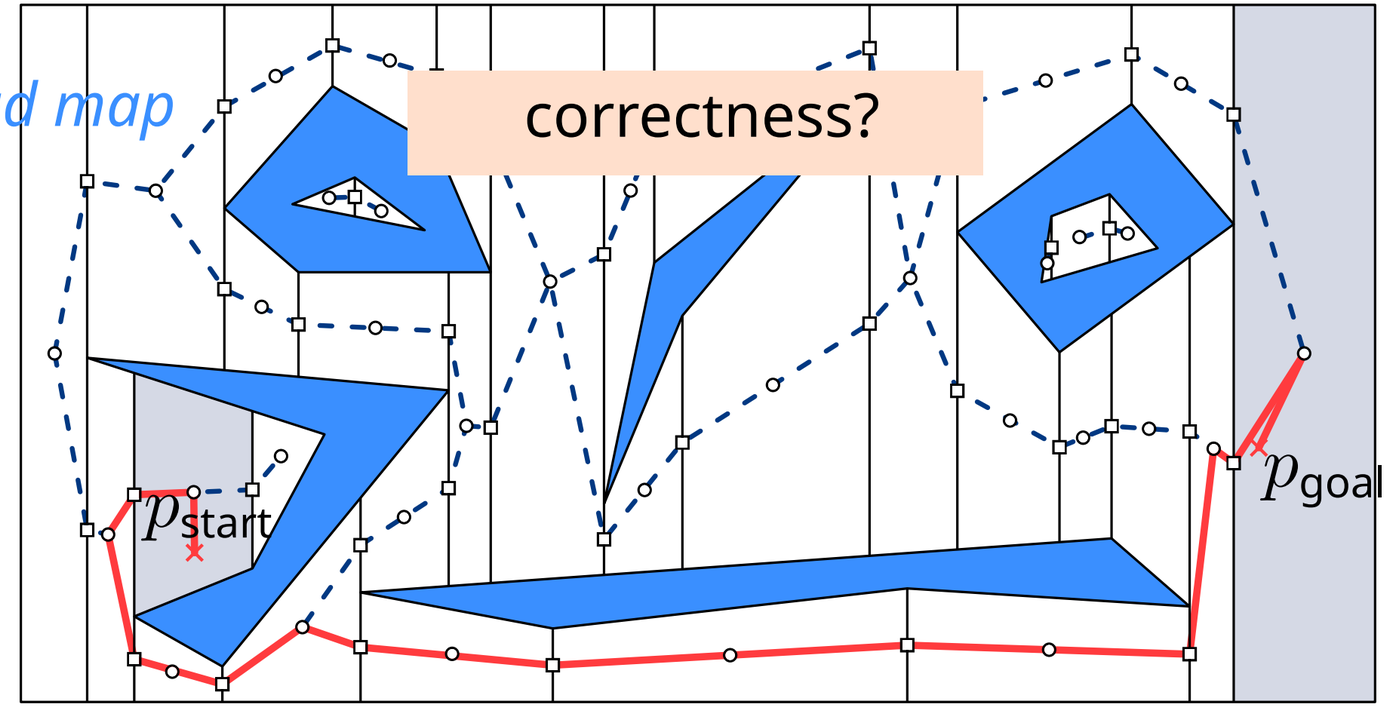


Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments
- locate start and goal
- find shortest path in G

road map

correctness?



Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments
- locate start and goal
- find shortest path in G

What is the overall running time of this approach?

A: Expected $\Theta(\log n)$

B: Expected $\Theta(n \log n)$

C: Expected $\Theta(n^2)$

Idea: shortest paths in graphs

- compute vertical decomposition
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments
- locate start and goal
- find shortest path in G

What is the overall running time of this approach?

A: Expected $\Theta(\log n)$

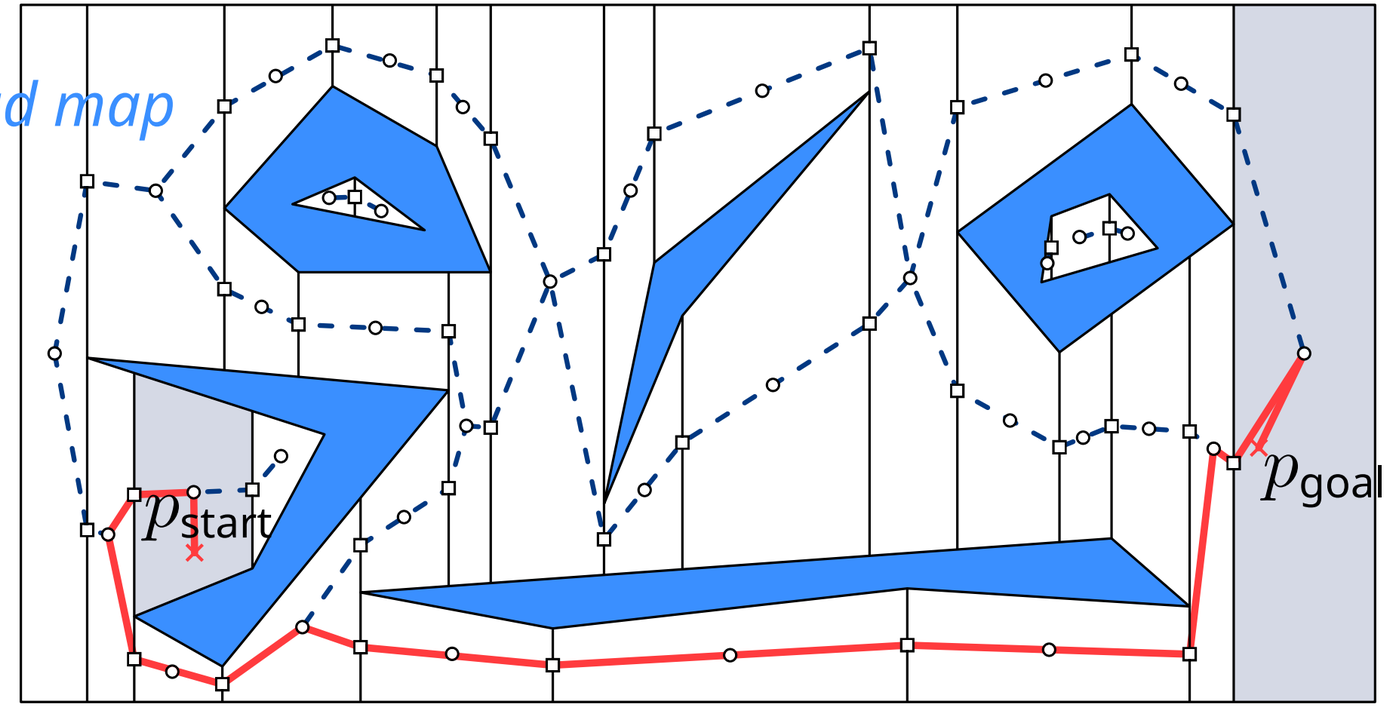
B: Expected $\Theta(n \log n)$

C: Expected $\Theta(n^2)$

Idea: shortest paths in graphs

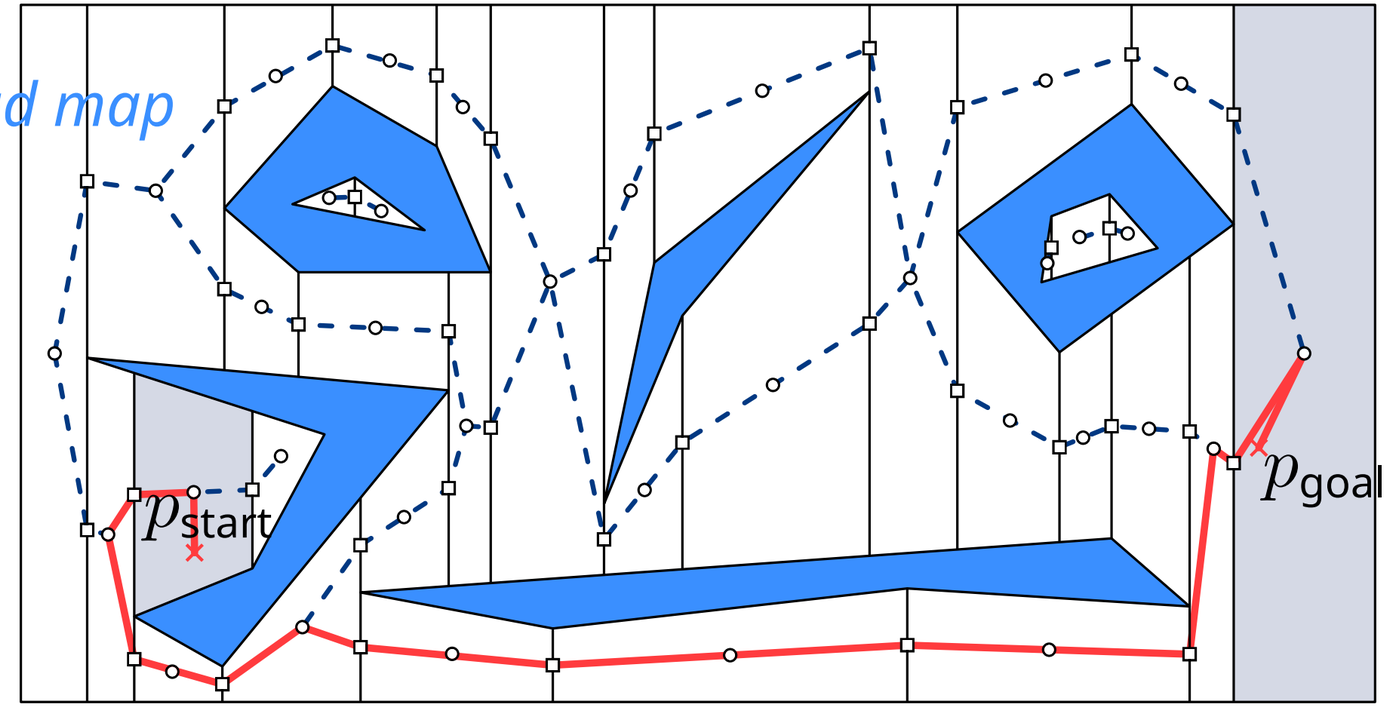
- compute vertical decomposition — expected $O(n \log n)$ time
- remove segments in obstacles
- graph G: vertices in trapezoids and vertical segments — size $O(n)$
- locate start and goal — expected $O(\log n)$ time
- find shortest path in G — $O(n)$ time (BFS)

road map



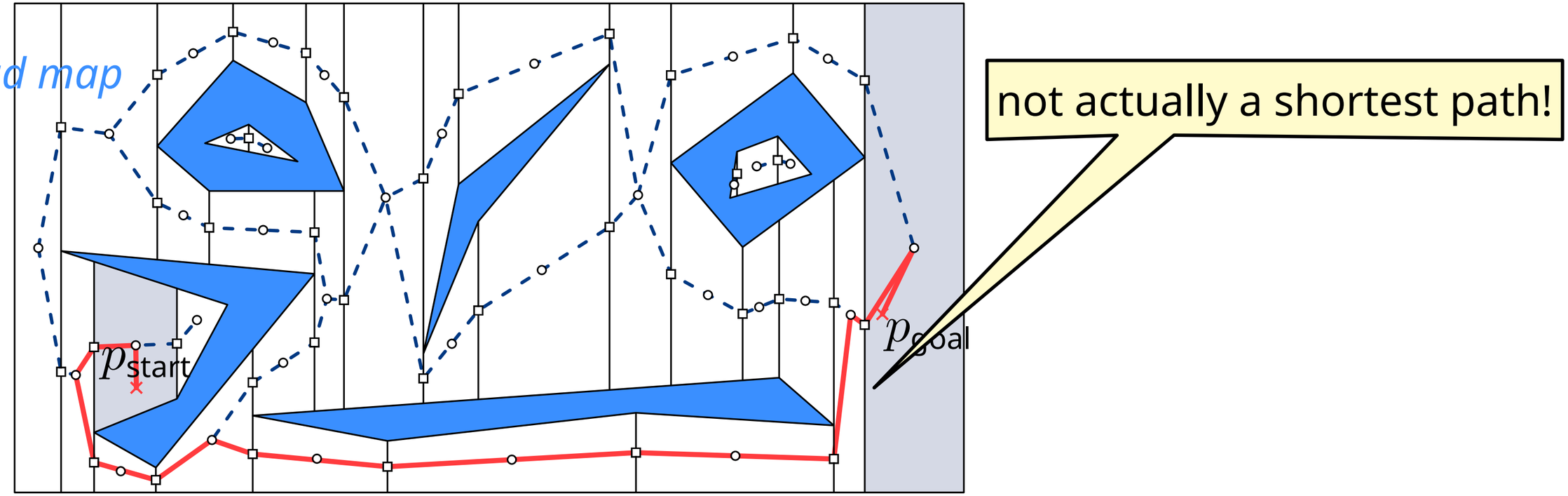
Idea: shortest paths in graphs

Theorem: Let S be a set of polygonal obstacles with n edges in total. We can preprocess S in $O(n \log n)$ expected time, such that between any start and goal position of a point robot a collision-free path can be computed in $O(n)$ time, if it exists.



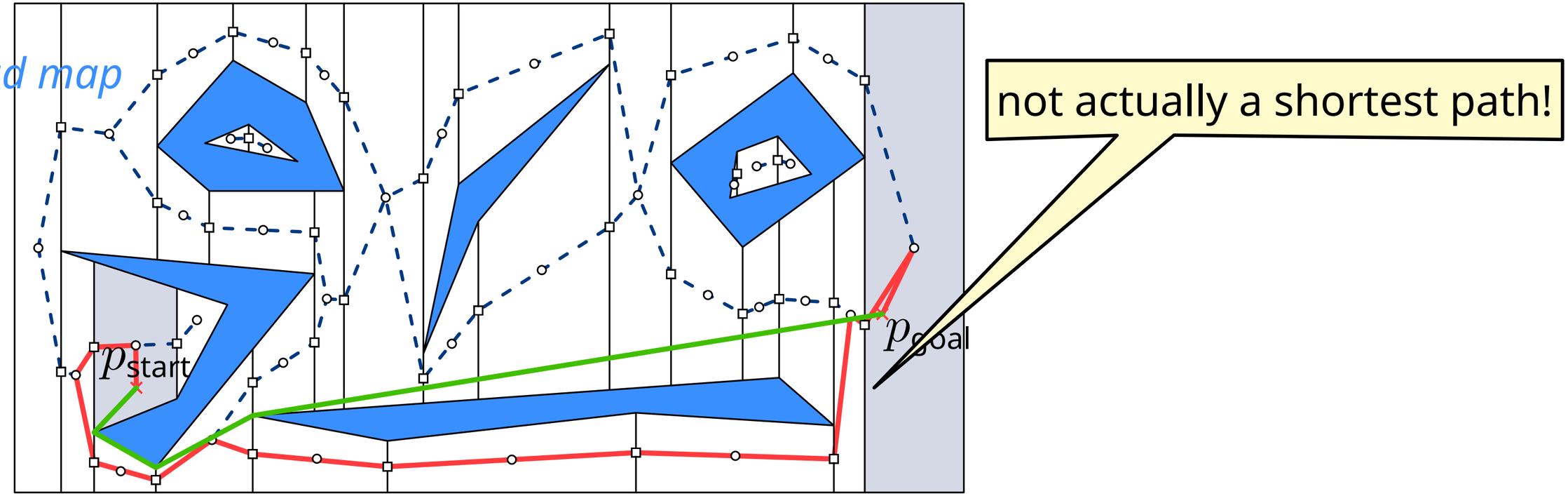
Idea: shortest paths in graphs

Theorem: Let S be a set of polygonal obstacles with n edges in total. We can preprocess S in $O(n \log n)$ expected time, such that between any start and goal position of a point robot a collision-free path can be computed in $O(n)$ time, if it exists.



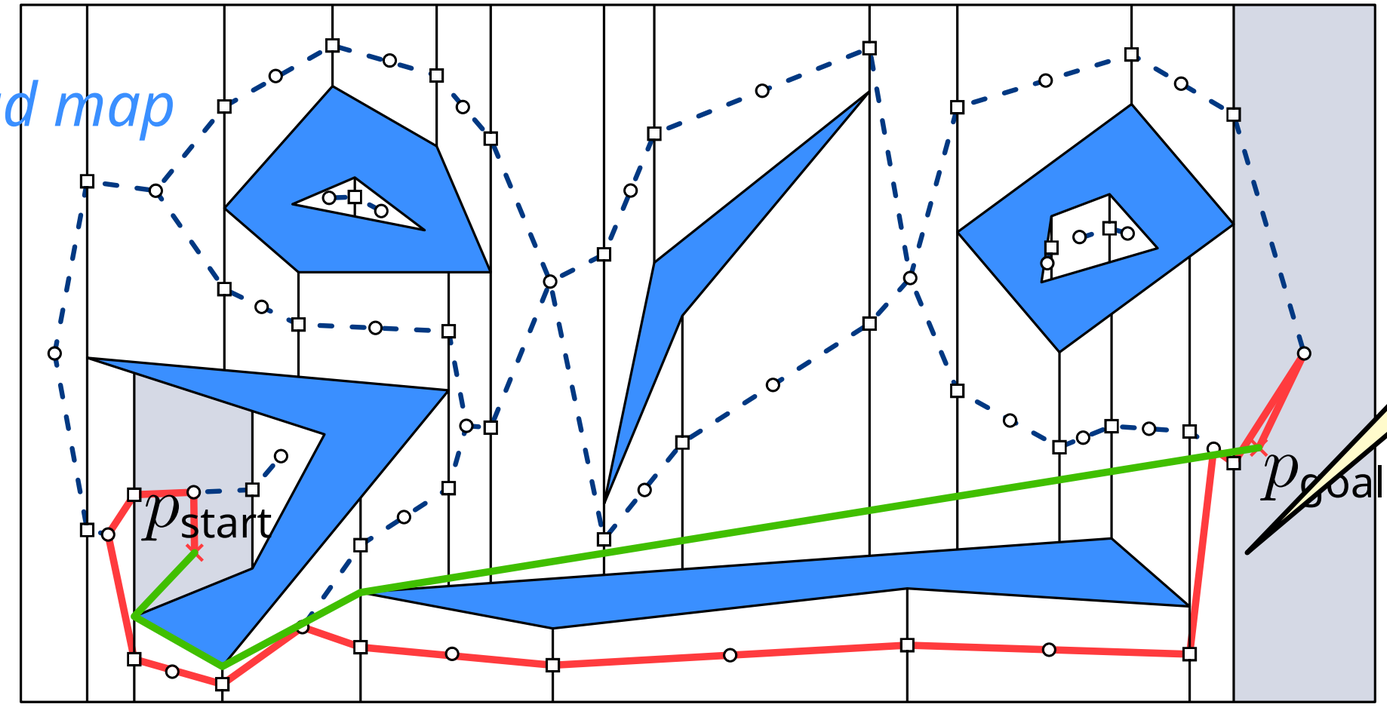
Idea: shortest paths in graphs

Theorem: Let S be a set of polygonal obstacles with n edges in total. We can preprocess S in $O(n \log n)$ expected time, such that between any start and goal position of a point robot a collision-free path can be computed in $O(n)$ time, if it exists.



Idea: shortest paths in graphs

Theorem: Let S be a set of polygonal obstacles with n edges in total. We can preprocess S in $O(n \log n)$ expected time, such that between any start and goal position of a point robot a collision-free path can be computed in $O(n)$ time, if it exists.



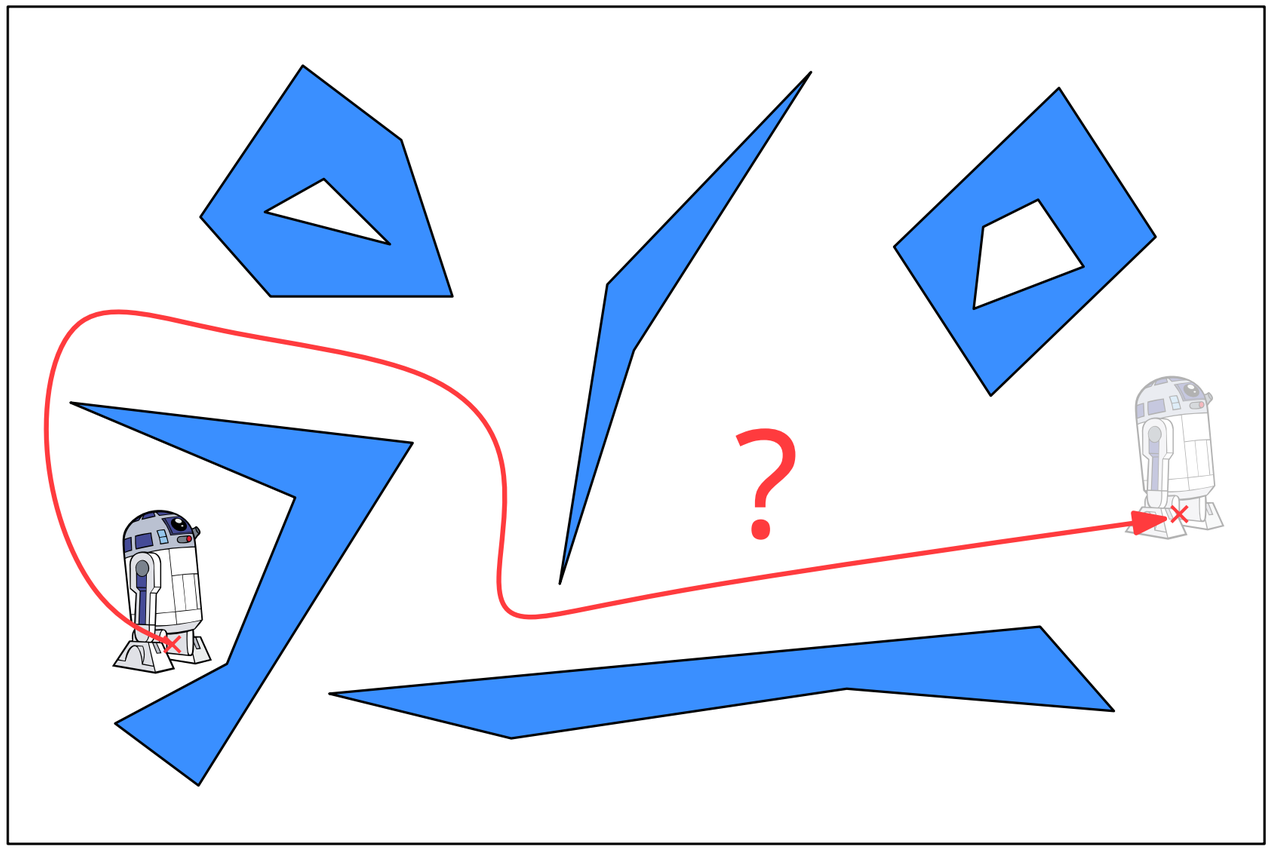
not actually a shortest path!

next: polygonal robot

Robot Motion Planning

configuration spaces

Motion planning for robots



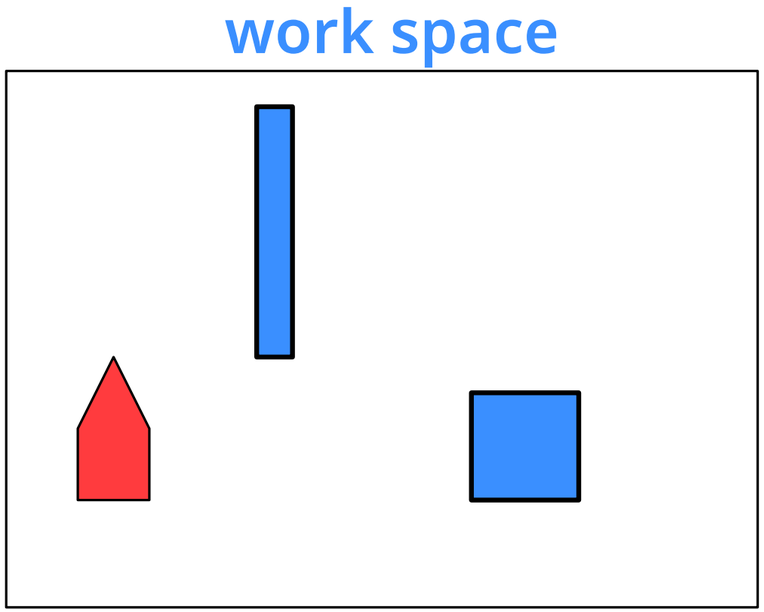
next

variants

- shape of robot **polygonal**
- shape of obstacles **polygonal**
- motion (e.g. rotation) of robot **translations**
- motion of obstacles **no motion**
- 2D or 3D **2D**

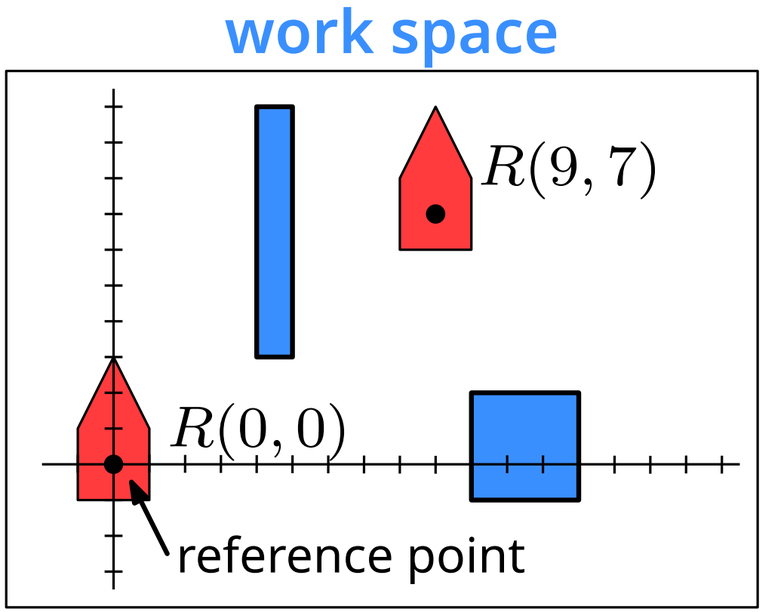
Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Configuration space



Configuration space

configuration: position of robot, can be described by translation vector

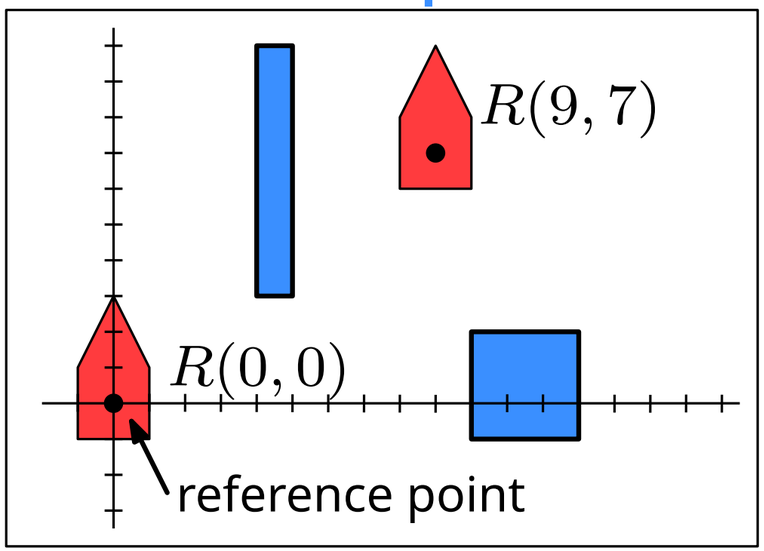


Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

work space



degrees of freedom

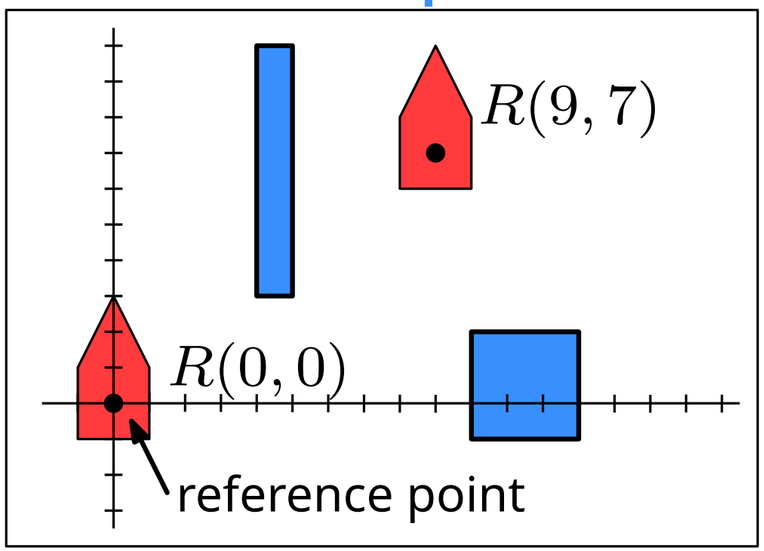
	translation	translation + rotation
2D		
3D		

Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

work space



degrees of freedom

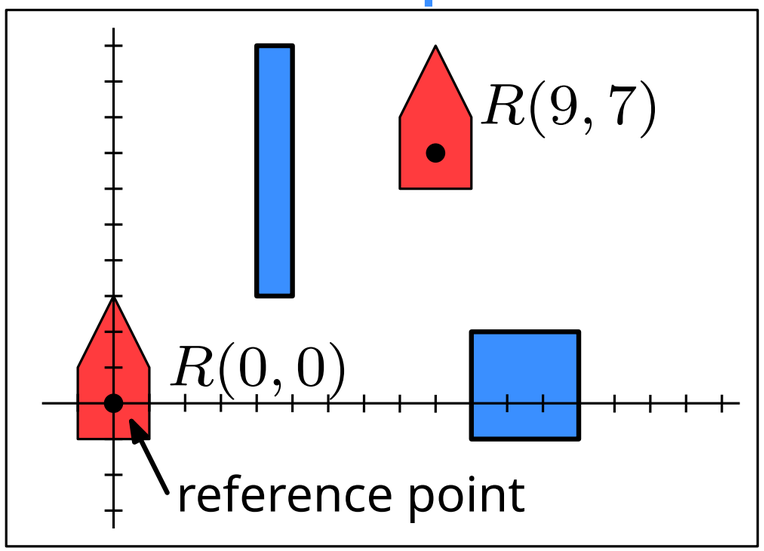
	translation	translation + rotation
2D	2	
3D		

Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

work space



degrees of freedom

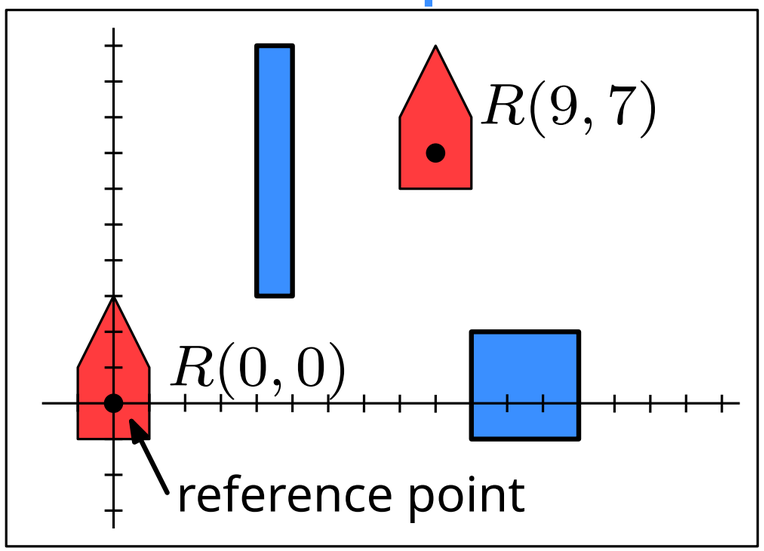
	translation	translation + rotation
2D	2	3
3D		

Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

work space



degrees of freedom

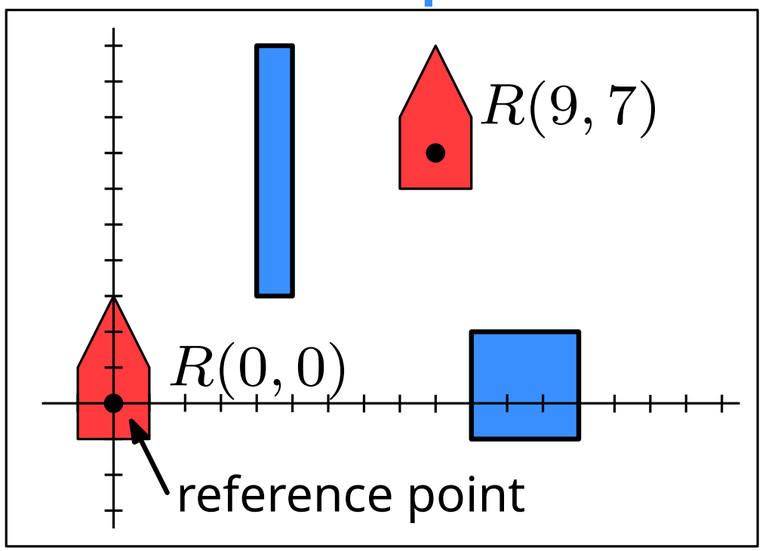
	translation	translation + rotation
2D	2	3
3D	3	

Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

work space

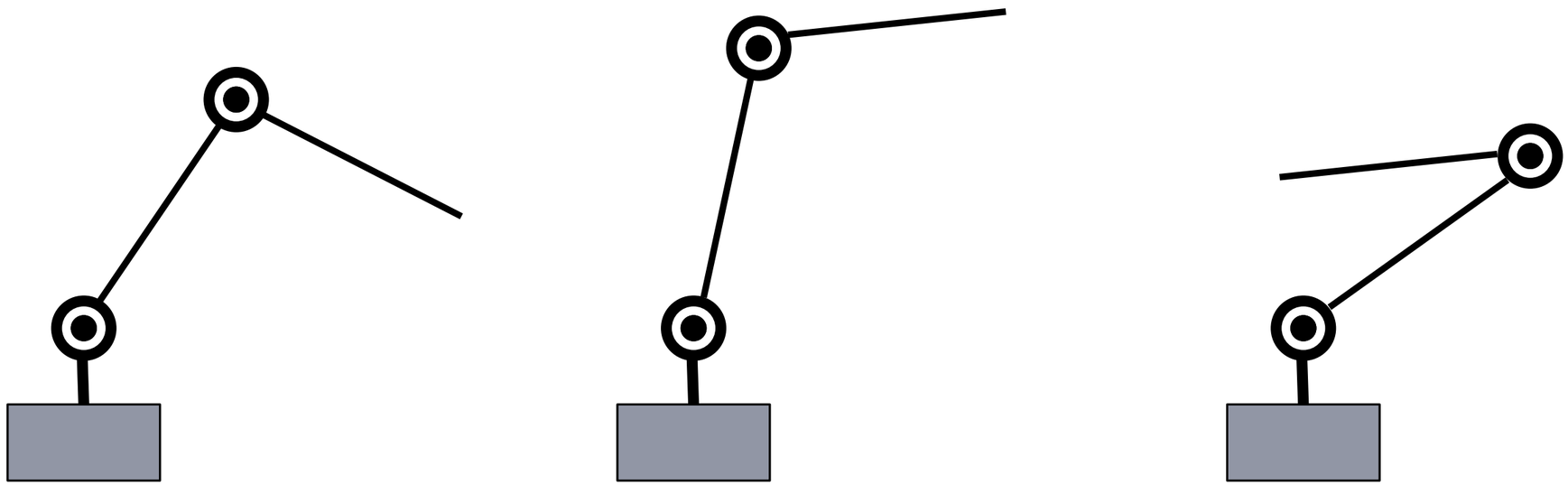


degrees of freedom

	translation	translation + rotation
2D	2	3
3D	3	6

Configuration space

How many degrees of freedom does the following robot have?



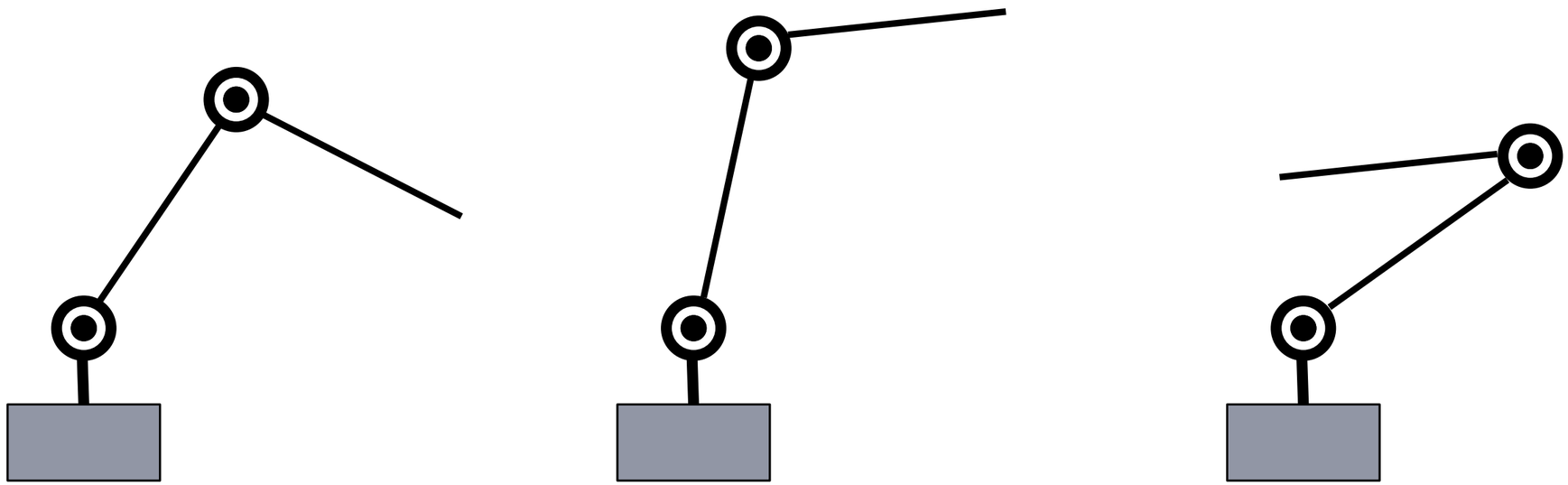
A: 2

B: 4

C: 6

Configuration space

How many degrees of freedom does the following robot have?



A: 2

B: 4

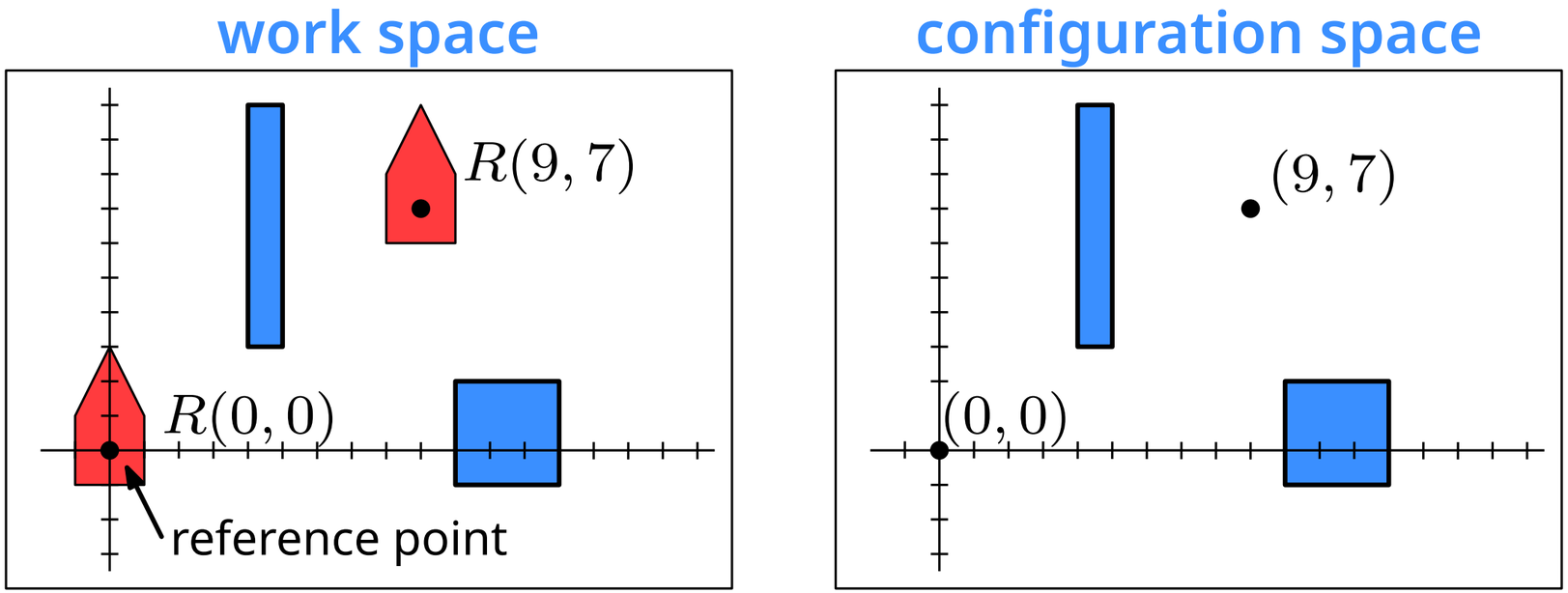
C: 6

Configuration space

configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

configuration space: the parameter space



Configuration space

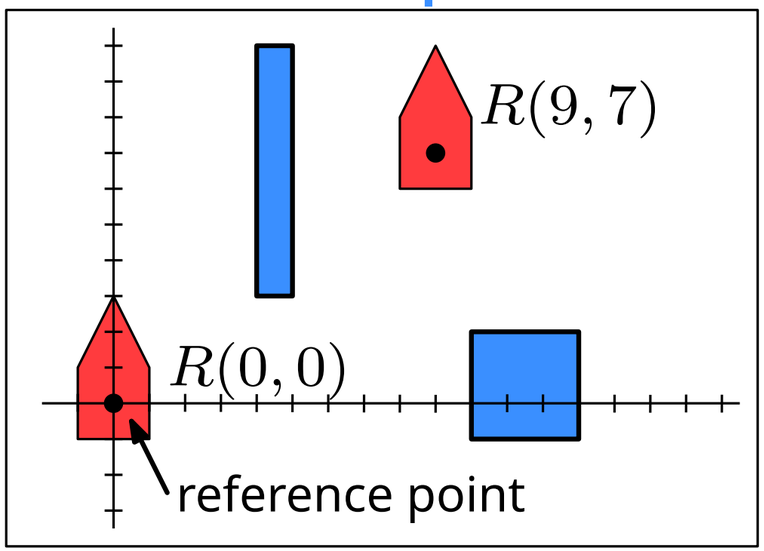
configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

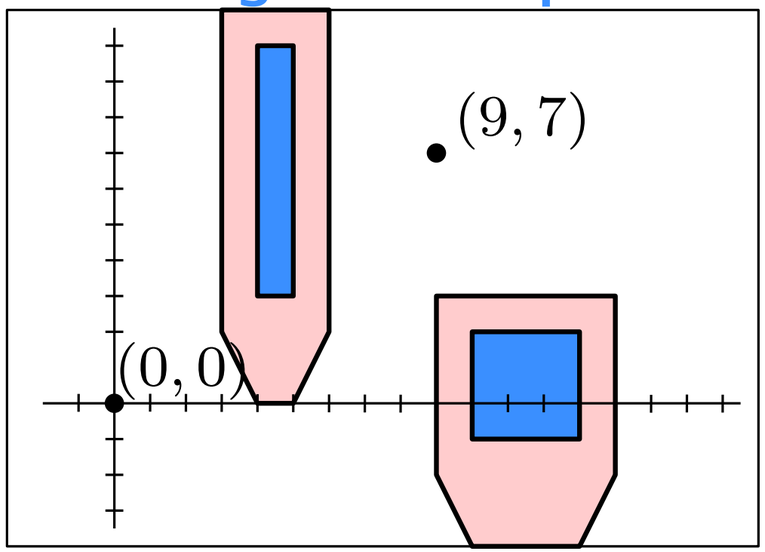
configuration space: the parameter space

if the robot intersects an obstacle, then the configuration is called forbidden, otherwise it is called free

work space



configuration space



Configuration space

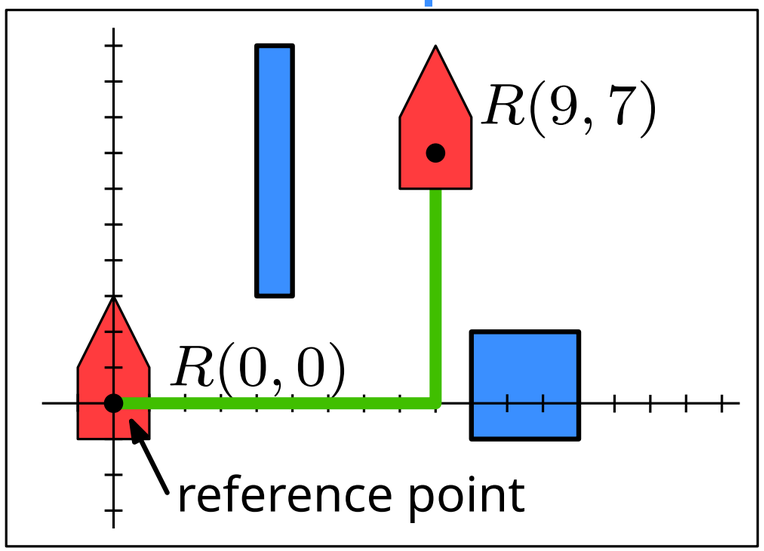
configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

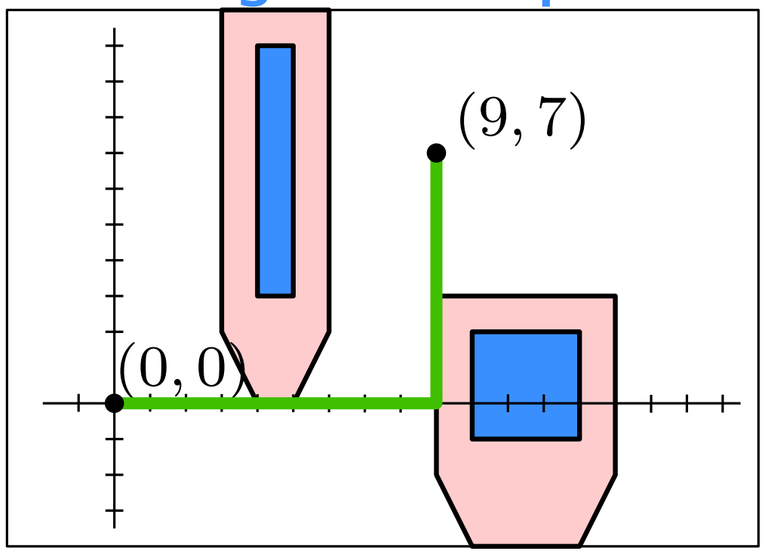
configuration space: the parameter space

if the robot intersects an obstacle, then the configuration is called forbidden, otherwise it is called free

work space



configuration space



collision-free path $\Leftrightarrow$ curve in free space

Configuration space

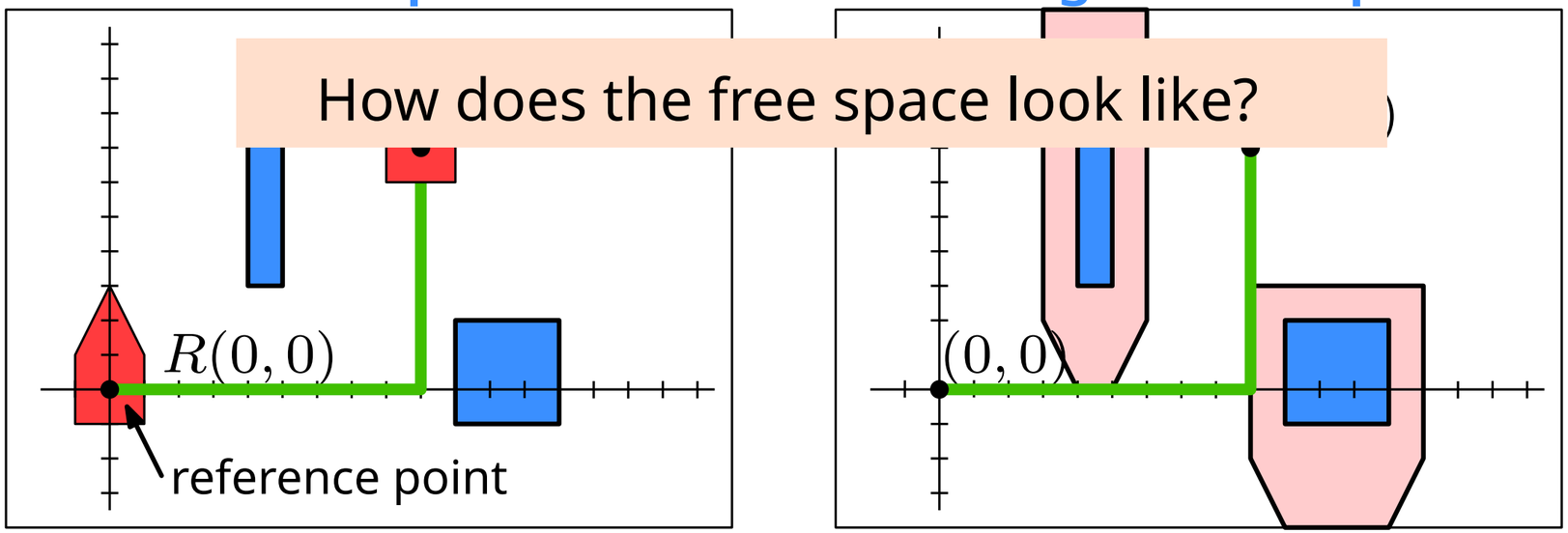
configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

configuration space: the parameter space

if the robot intersects an obstacle, then the configuration is called forbidden, otherwise it is called free

work space **configuration space**



collision-free path $\Leftrightarrow$ curve in free space

Configuration space

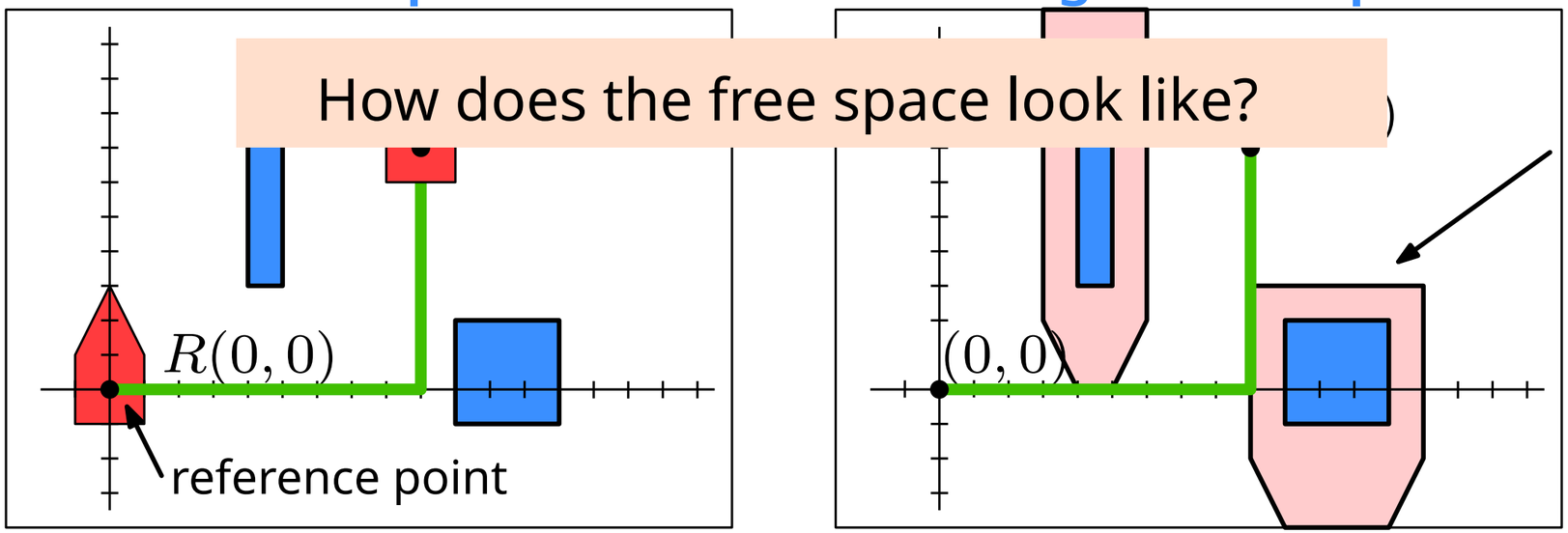
configuration: position of robot, can be described by translation vector

degrees of freedom: number of parameters needed to specify the configuration

configuration space: the parameter space

if the robot intersects an obstacle, then the configuration is called forbidden, otherwise it is called free

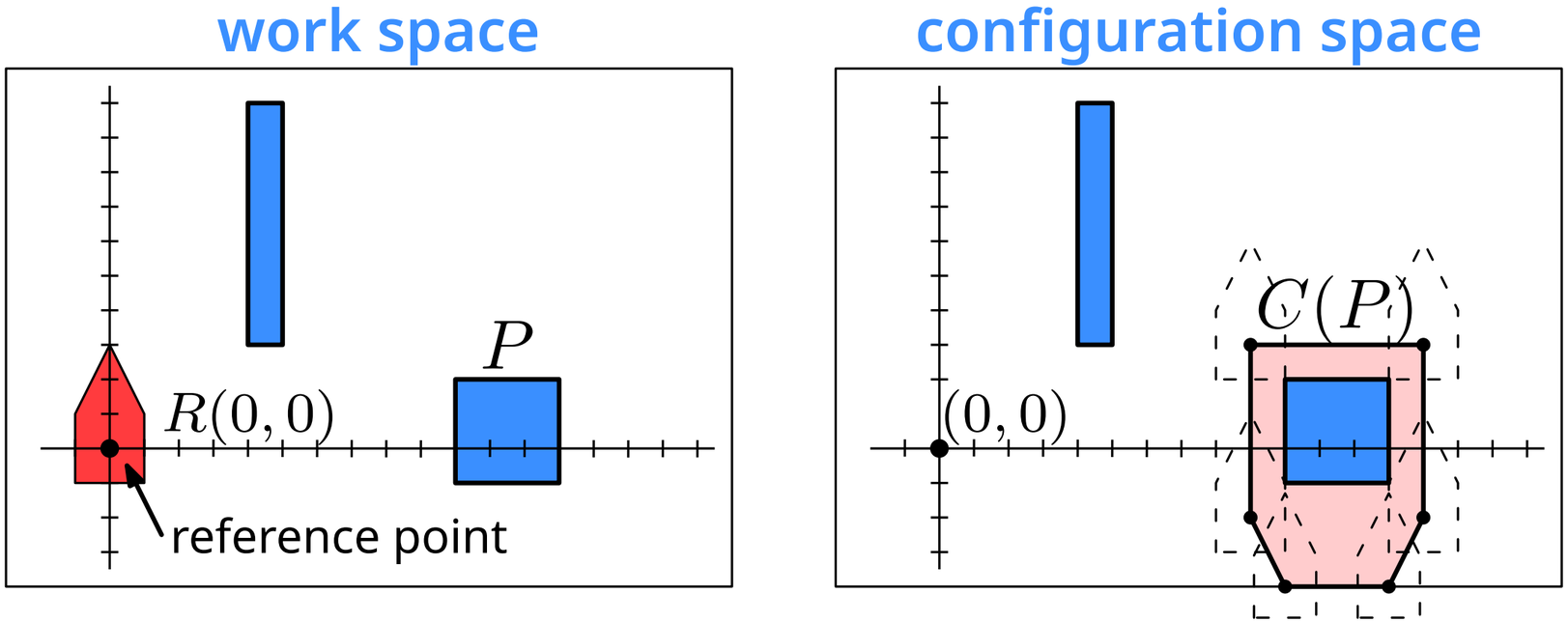
work space **configuration space**



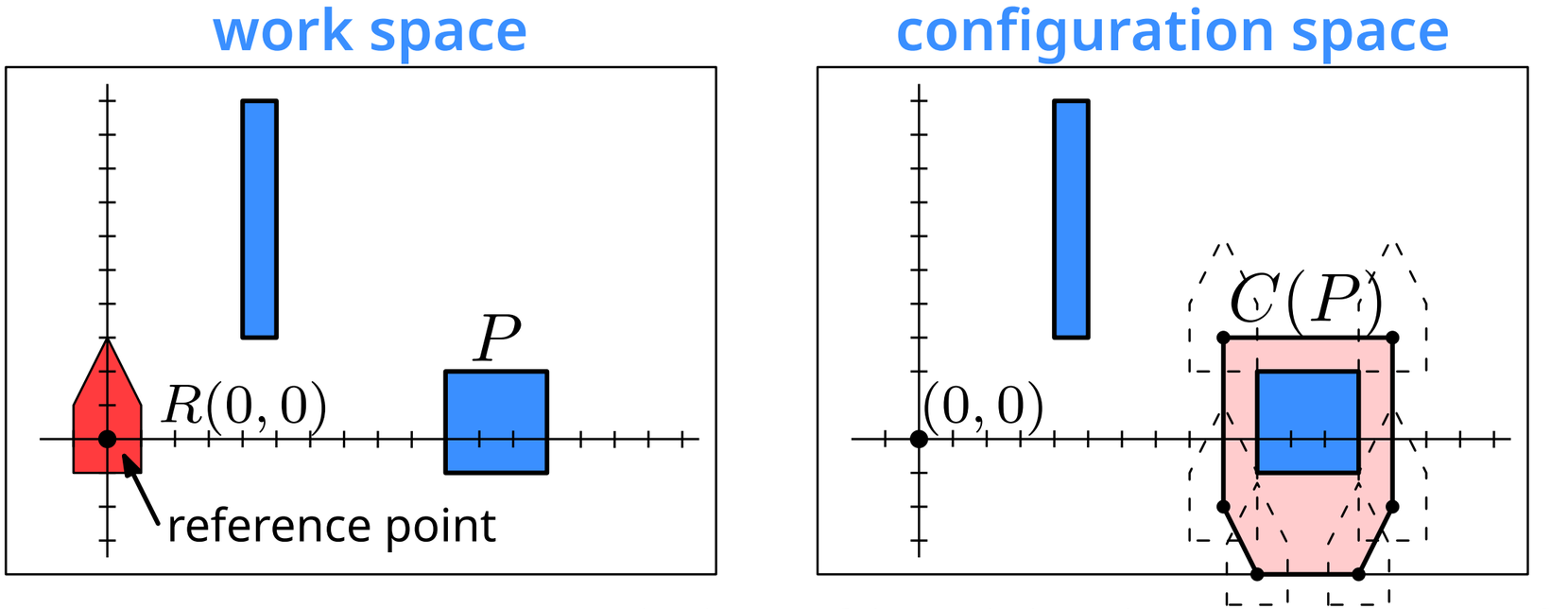
complement of the union of the forbidden spaces of all obstacles

collision-free path $\Leftrightarrow$ curve in free space

Configuration space



Configuration space

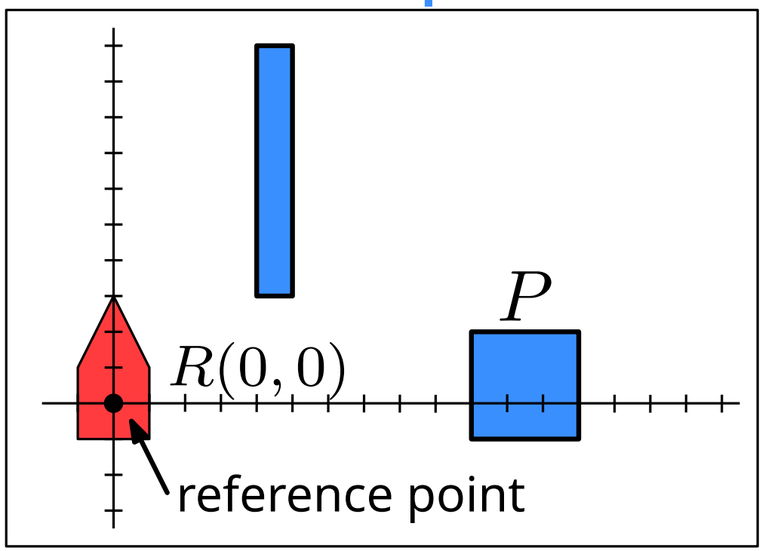


for an obstacle P let $C(P) = \{(x, y) \mid R(x, y) \cap P \neq \varnothing\}$

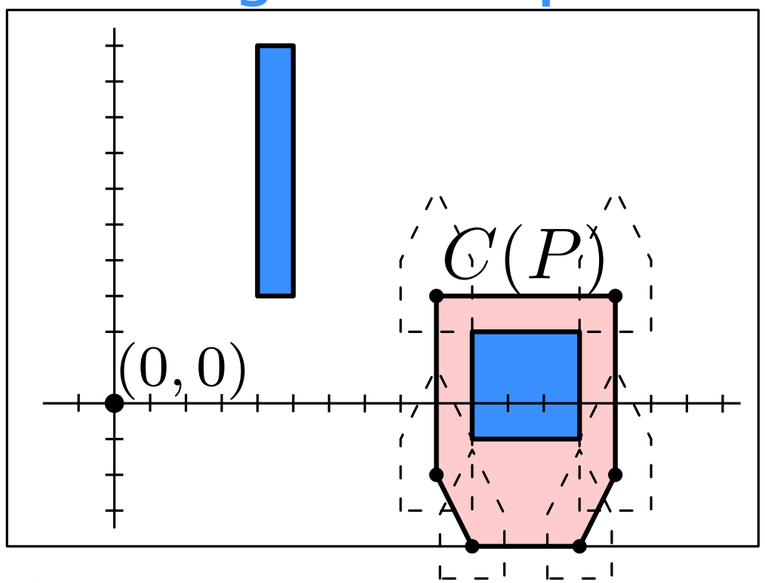
Configuration space

What is the forbidden space $C(P)$ of obstacle P?
How can we compute it?

work space



configuration space



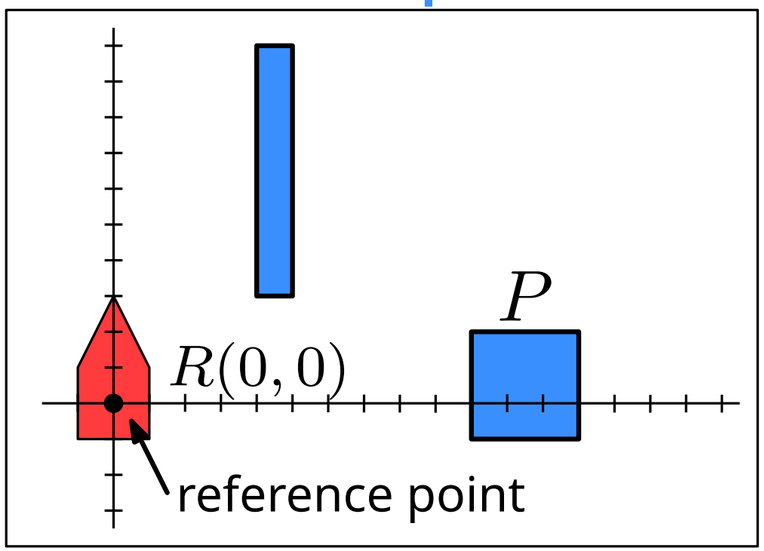
for an obstacle P let $C(P) = \{(x, y) \mid R(x, y) \cap P \neq \varnothing\}$

Configuration space

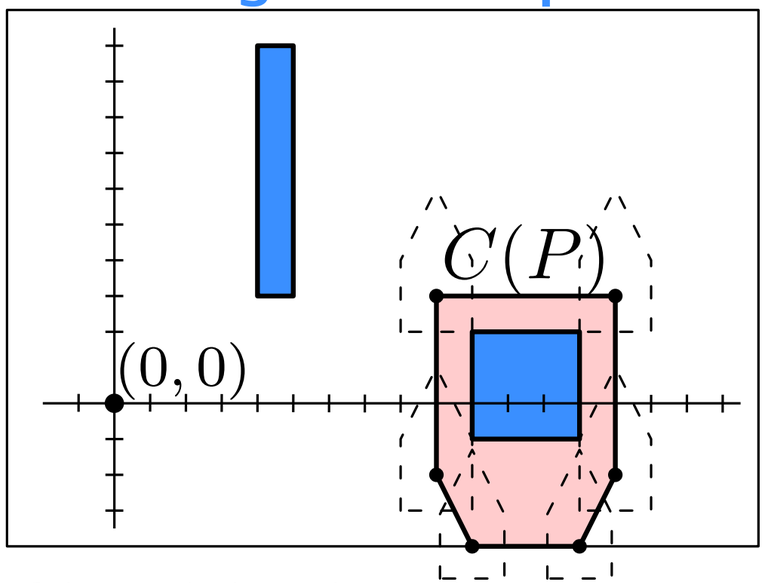
What is the forbidden space $C(P)$ of obstacle P?
How can we compute it?

Next: Minkowski sums

work space



configuration space



for an obstacle P let $C(P) = \{(x,y) \mid R(x,y) \cap P \neq \varnothing\}$

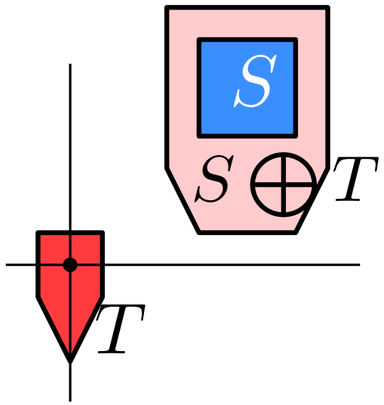
Robot Motion Planning

Minkowski sums

Minkowski sums

For S and $T \subseteq \mathbb{R}^2$ let

$$S \bigoplus T := \{p + q \mid p \in S, q \in T\}$$

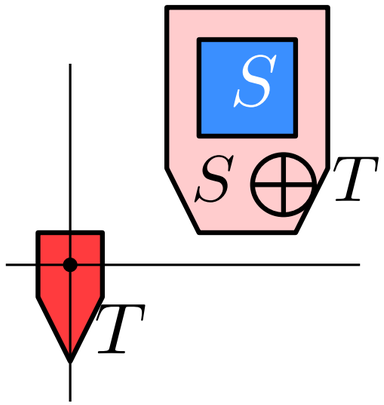


Minkowski sums

For S and $T \subseteq \mathbb{R}^2$ let

$$S \bigoplus T := \{p + q \mid p \in S, q \in T\}$$

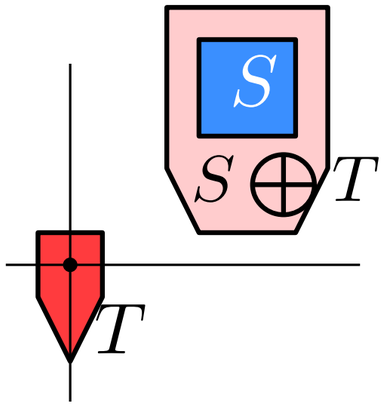
Inversion of S is $-S = \{-p \mid p \in S\}$ (reflect about origin)



Minkowski sums

For S and $T \subseteq \mathbb{R}^2$ let

$$S \bigoplus T := \{p + q \mid p \in S, q \in T\}$$



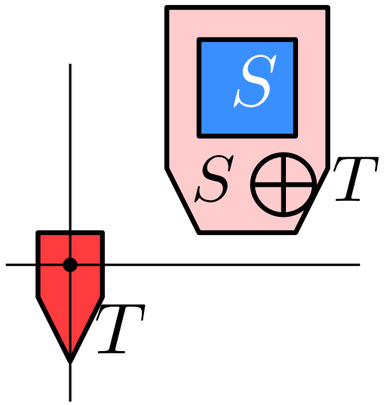
Inversion of S is $-S = \{-p \mid p \in S\}$ (reflect about origin)

Theorem: Let R be a planar, translating robot and P an obstacle.
Then $C(P) = P \bigoplus -R(0,0)$.

Minkowski sums

For S and $T \subseteq \mathbb{R}^2$ let

$$S \bigoplus T := \{p + q \mid p \in S, q \in T\}$$

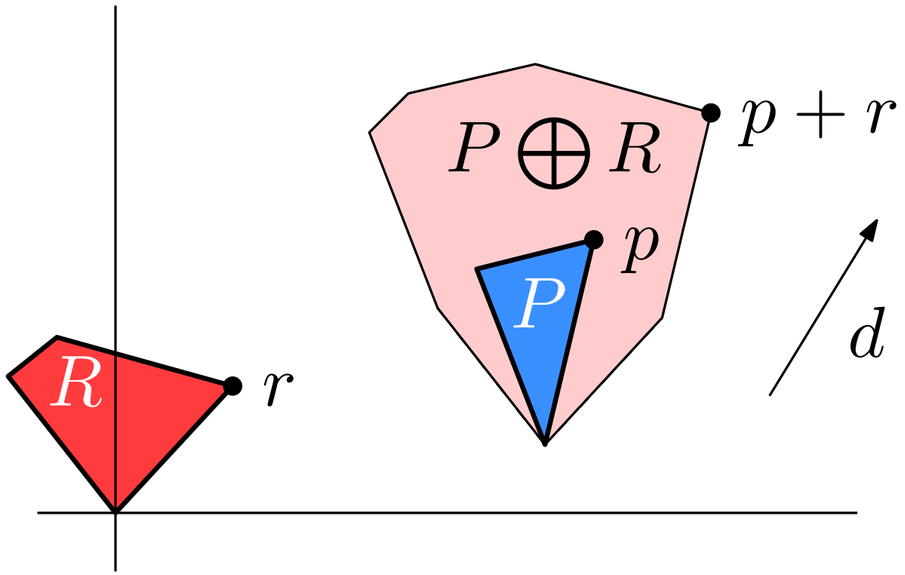


Inversion of S is $-S = \{-p \mid p \in S\}$ (reflect about origin)

Theorem: Let R be a planar, translating robot and P an obstacle.
Then $C(P) = P \bigoplus -R(0,0)$. $(C(P) = \{(x,y) \mid R(x,y) \cap P \neq \varnothing\})$

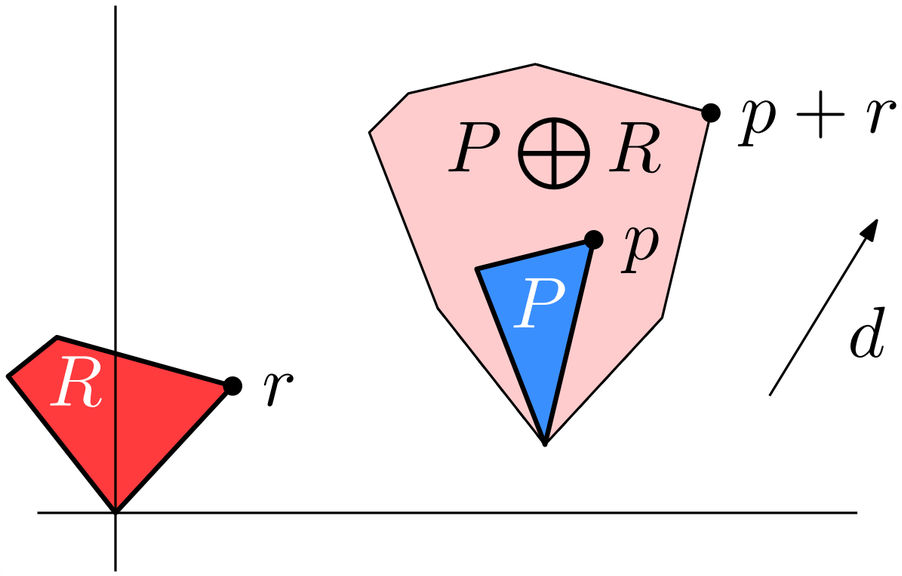
Minkowski sums

Observation: Let P and Q be objects in the plane. An extreme point of $P \bigoplus Q$ in direction d is the sum of extreme points in direction d of P and Q.



Minkowski sums

Observation: Let P and Q be objects in the plane. An extreme point of $P \bigoplus Q$ in direction d is the sum of extreme points in direction d of P and Q.



How many edges does $P \bigoplus Q$ have at most if P and Q are convex polygons with m and n edges, respectively?

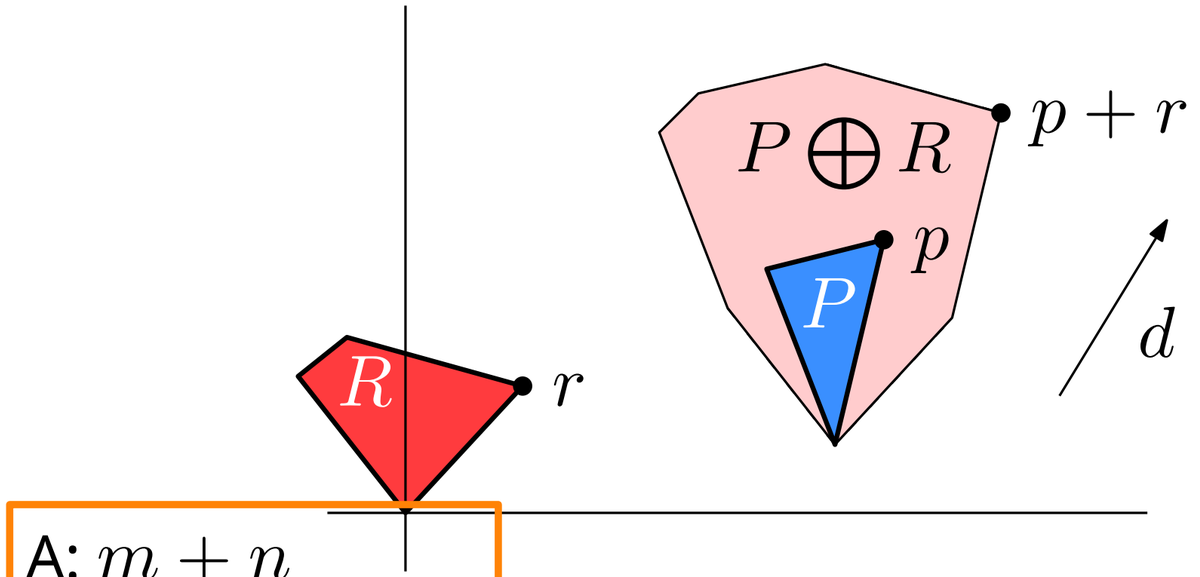
A: $m + n$

B: $m \cdot n$

C: $\max(m, n)$

Minkowski sums

Observation: Let P and Q be objects in the plane. An extreme point of $P \bigoplus Q$ in direction d is the sum of extreme points in direction d of P and Q.



How many edges does $P \bigoplus Q$ have at most if P and Q are convex polygons with m and n edges, respectively?

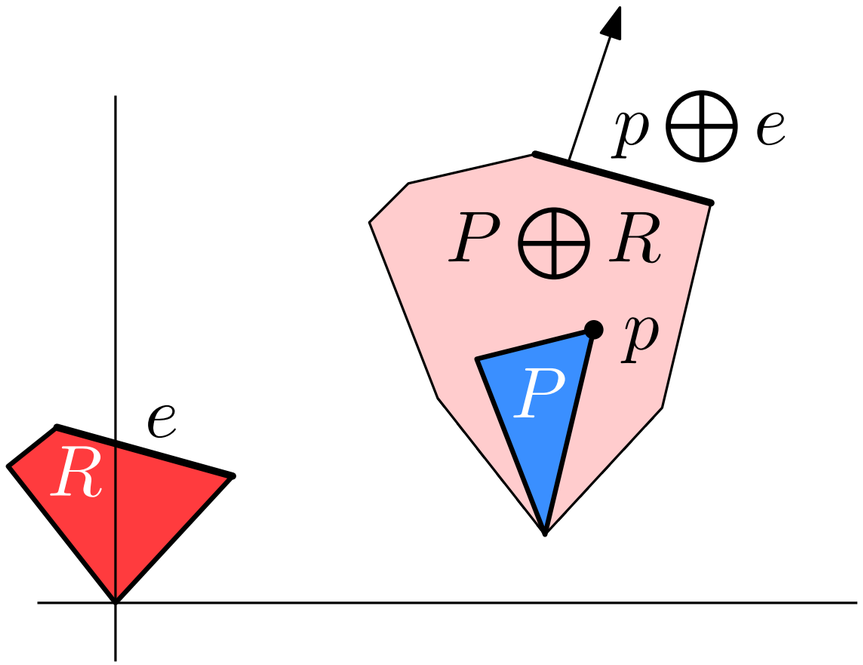
A: $m + n$

B: $m \cdot n$

C: $\max(m, n)$

Minkowski sums

Observation: Let P and Q be objects in the plane. An extreme point of $P \bigoplus Q$ in direction d is the sum of extreme points in direction d of P and Q.



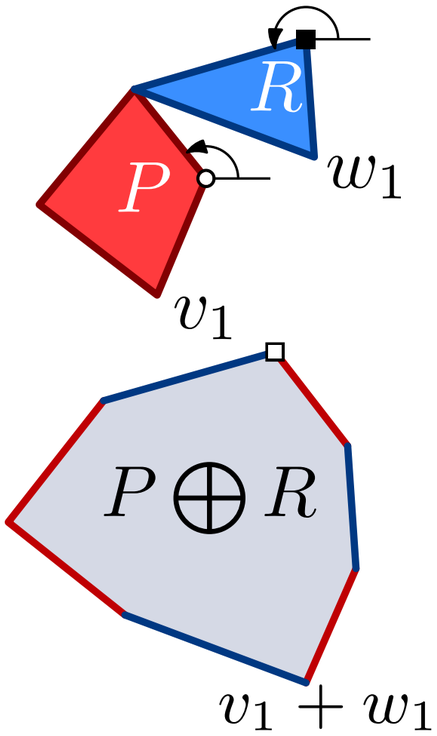
Theorem: Let P and Q be convex polygons with n and m edges, respectively. Then $P \bigoplus Q$ is a convex polygon with at most $m + n$ edges.

Computing Minkowski sums

Theorem: The Minkowski sum of two convex polygons with n and m edges can be computed in $O(m + n)$ time.

Computing Minkowski sums

Theorem: The Minkowski sum of two convex polygons with n and m edges can be computed in $O(m+n)$ time.



Computing Minkowski sums

Theorem: The Minkowski sum of two convex polygons with n and m edges can be computed in $O(m+n)$ time.

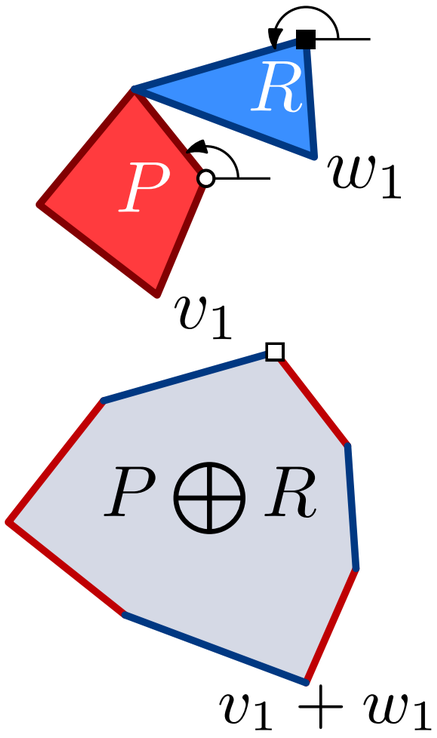
Algorithm: linear scan over vertices along extreme direction

Input: two convex polygons P and R with vertices $v_1, \ldots, v_n$ and $w_1, \ldots, w_m$ in ccw order starting with smallest y

Output: $P \bigoplus R$

1: process edges of P and R in (merged) order of there direction (angle with positive x-axis)

2: while doing so: add $v_i + w_j$ as vertex to $P \bigoplus R$, where v_i and w_j are the endpoints of the last edges processed from P and R, respectively

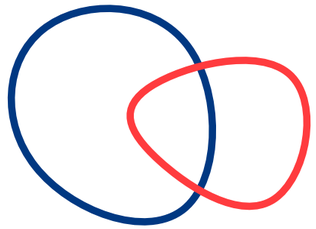


Robot Motion Planning

Unions of Minkowski sums

Pseudodiscs

Definition: Two planar objects P, Q are called a pair of pseudodiscs if $\delta P \cap interior(Q)$ and $\delta Q \cap interior(P)$ are connected.

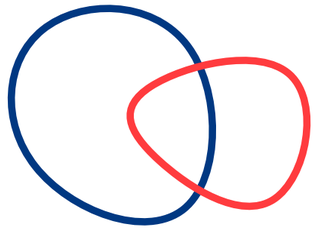


A set of objects is called a set of pseudodiscs if every pair is a pair of pseudodiscs.

A point $p = \delta P \cap \delta Q$ is called boundary crossing if δP crosses in p from the inside to the outside of Q.

Pseudodiscs

Definition: Two planar objects P, Q are called a pair of pseudodiscs if $\delta P \cap interior(Q)$ and $\delta Q \cap interior(P)$ are connected.



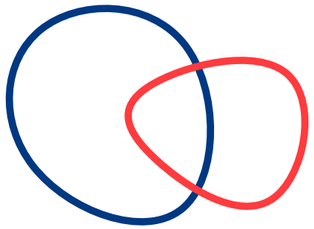
A set of objects is called a set of pseudodiscs if every pair is a pair of pseudodiscs.

A point $p = \delta P \cap \delta Q$ is called boundary crossing if δP crosses in p from the inside to the outside of Q.

Theorem: Let P, Q be convex polygons with disjoint interiors, and R another convex polygon. Then $P \bigoplus R$ and $Q \bigoplus R$ are a pair of pseudodiscs.

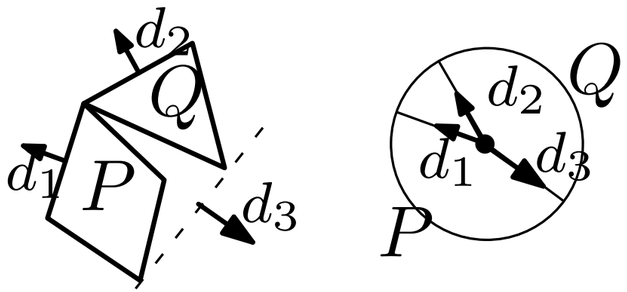
Pseudodiscs

Definition: Two planar objects P, Q are called a pair of pseudodiscs if $\delta P \cap interior(Q)$ and $\delta Q \cap interior(P)$ are connected.



A set of objects is called a set of pseudodiscs if every pair is a pair of pseudodiscs.

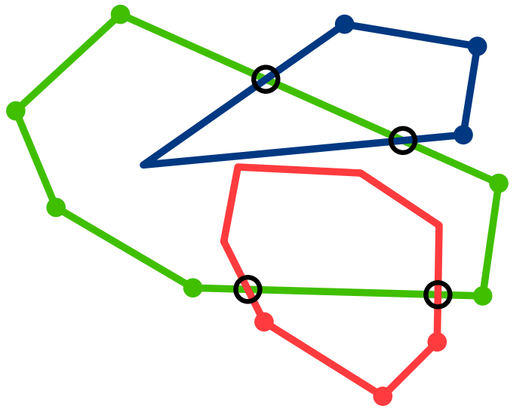
A point $p = \delta P \cap \delta Q$ is called boundary crossing if δP crosses in p from the inside to the outside of Q.



Observation: Let P, Q be convex polygons with disjoint interiors. If P is more extreme than Q in directions d_1, d_2 then P is more extreme than Q in all direction between d_1 and d_2 or between d_2 and d_1.

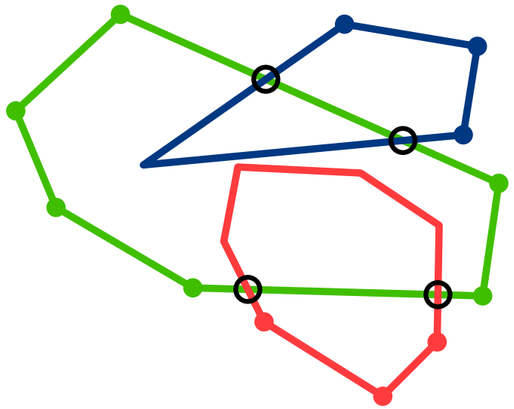
Complexity

Theorem: The union of a set of convex polygonal pseudodiscs with n vertices has at most $2n$ vertices.



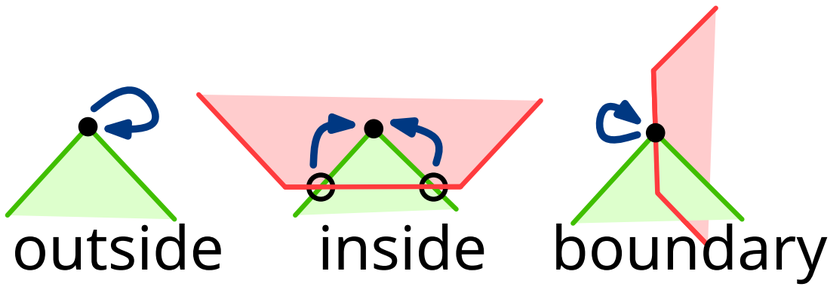
Complexity

Theorem: The union of a set of convex polygonal pseudodiscs with n vertices has at most $2n$ vertices.



Proof: Move 1 charge from every vertex of the union s.t. every vertex from the input receives charge ≤ 2

charge a vertex to itself
and a crossing to an inner vertex of that edge



Quiz + Summary, so far

Algorithm for convex robots (translations):

- triangulate obstacles if not convex (Lec. 3)
- compute $C(P)$ for every obstacle P
- compute union of obstacles (div&conq, pairwise union using "map overlay" (Lec. 2)
- compute vertical decomposition (Lec. 4) of complement = free space
- compute shortest path in the road map graph

Quiz + Summary, so far

Algorithm for convex robots (translations):

- triangulate obstacles if not convex (Lec. 3)
- compute $C(P)$ for every obstacle P
- compute union of obstacles (div&conq, pairwise union using "map overlay" (Lec. 2)
- compute vertical decomposition (Lec. 4) of complement = free space
- compute shortest path in the road map graph

What is the running time of this algorithm?

A: $O(n \log n)$

B: $O(n \log^2 n)$

C: $O(n^2 \log n)$

Quiz + Summary, so far

Algorithm for convex robots (translations):

- triangulate obstacles if not convex (Lec. 3)
- compute $C(P)$ for every obstacle P
- compute union of obstacles (div&conq, pairwise union using "map overlay" (Lec. 2)
- compute vertical decomposition (Lec. 4) of complement = free space
- compute shortest path in the road map graph

What is the running time of this algorithm?

A: $O(n \log n)$

B: $O(n \log^2 n)$

C: $O(n^2 \log n)$

Quiz + Summary, so far

Algorithm for convex robots (translations):

- triangulate obstacles if not convex (Lec. 3)
- compute $C(P)$ for every obstacle P
- compute union of obstacles (div&conq, pairwise union using "map overlay" (Lec. 2)
- compute vertical decomposition (Lec. 4) of complement = free space
- compute shortest path in the road map graph

Theorem: Let R be a convex robot of constant complexity translating among a set S of disjoint polygonal obstacles with n edges in total. We can preprocess S in $O(n \log^2 n)$ expected time, such that between any start and goal position a collision-free path for R, if it exists, can be computed in $O(n)$ time.

Complexity of Minkowski sums, continued

Theorem: Let P, R be polygons with n and m edges. Then $P \bigoplus R$ is a polygon with

- $O(m+n)$ edges, if P, R convex
- $O(mn)$ edges, if R convex, P not convex
- $O(m^2n^2)$ edges, if P, R not convex.

These bounds are achieved in the worst case.

Complexity of Minkowski sums, continued

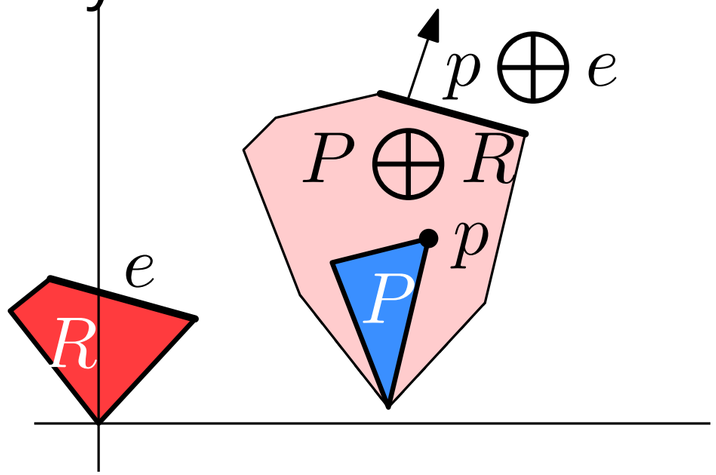
Theorem: Let P, R be polygons with n and m edges. Then $P \bigoplus R$ is a polygon with

- $O(m+n)$ edges, if P, R convex
- $O(mn)$ edges, if R convex, P not convex
- $O(m^2n^2)$ edges, if P, R not convex.

These bounds are achieved in the worst case.

Proof: both convex, then we know:

boundary = extreme vertices = sum of extreme vertices



Complexity of Minkowski sums, continued

Theorem: Let P, R be polygons with n and m edges. Then $P \bigoplus R$ is a polygon with

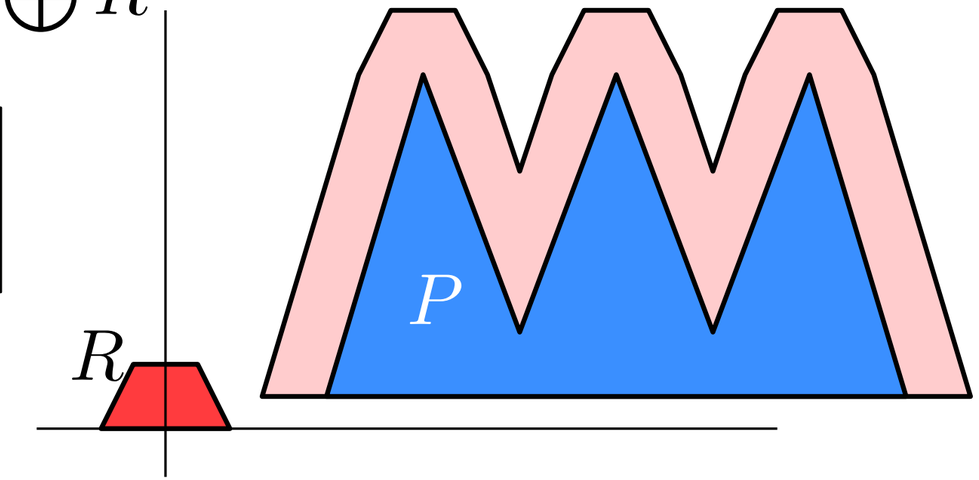
- $O(m+n)$ edges, if P, R convex
- $O(mn)$ edges, if R convex, P not convex
- $O(m^2n^2)$ edges, if P, R not convex.

These bounds are achieved in the worst case.

Proof: triangulate P in $n-2$ triangles $t_1, \ldots, t_{n-2}$, then

$$P \bigoplus R = \bigcup_i^{n-2} t_i \bigoplus R$$

convex with $m+3$ edges
set of pseudodiscs

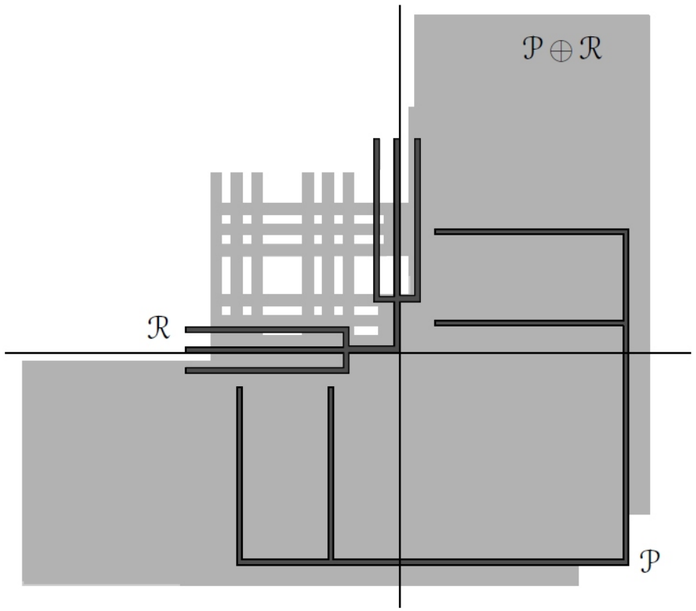


Complexity of Minkowski sums, continued

Theorem: Let P, R be polygons with n and m edges. Then $P \bigoplus R$ is a polygon with

- $O(m+n)$ edges, if P, R convex
- $O(mn)$ edges, if R convex, P not convex
- $O(m^2n^2)$ edges, if P, R not convex.

These bounds are achieved in the worst case.



Proof:

triangulate P and R in $n-2$ and $m-2$ triangles resp., then

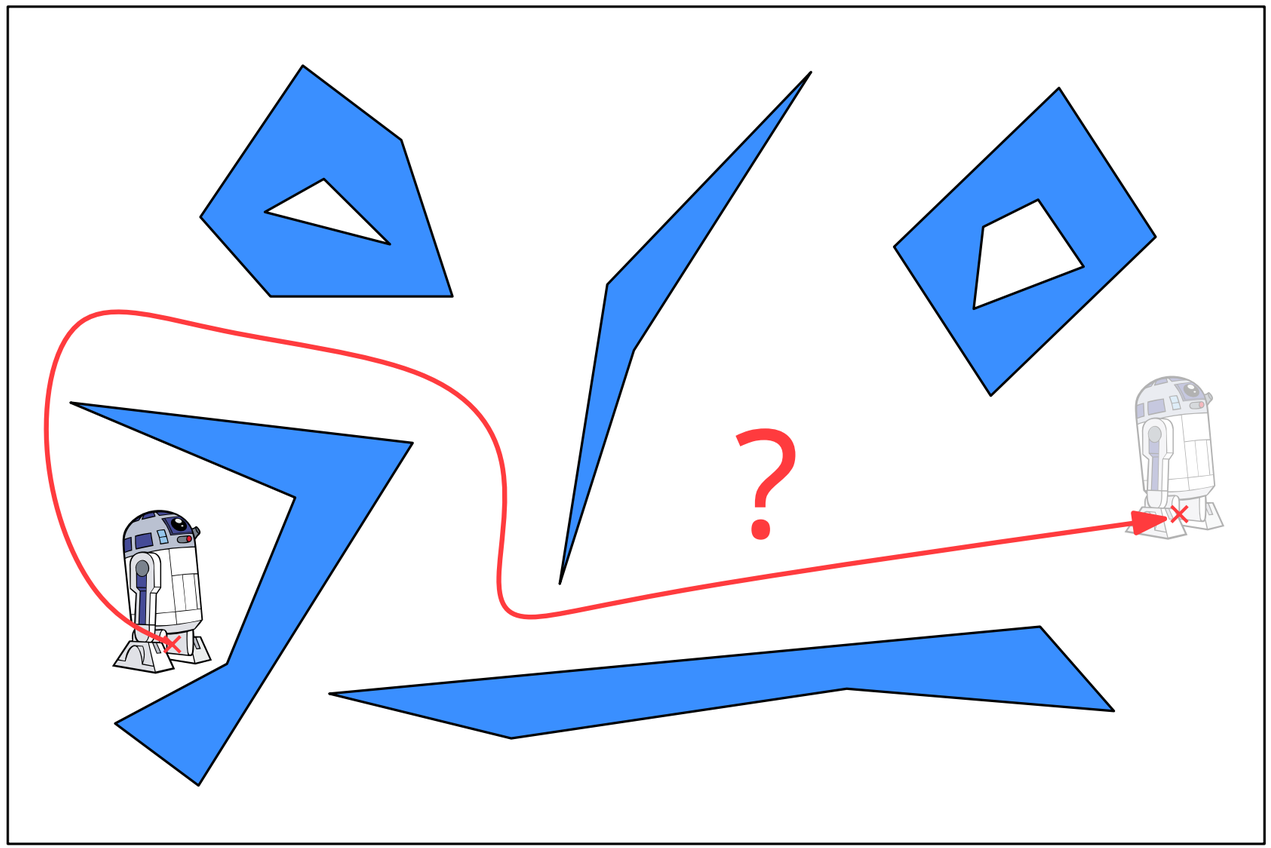
$$P \bigoplus R = \bigcup_{i=1}^{n-2} \bigcup_{j=1}^{m-2} t_i \bigoplus u_j$$

constant complexity

Robot Motion Planning

translations + rotations

Motion planning for robots



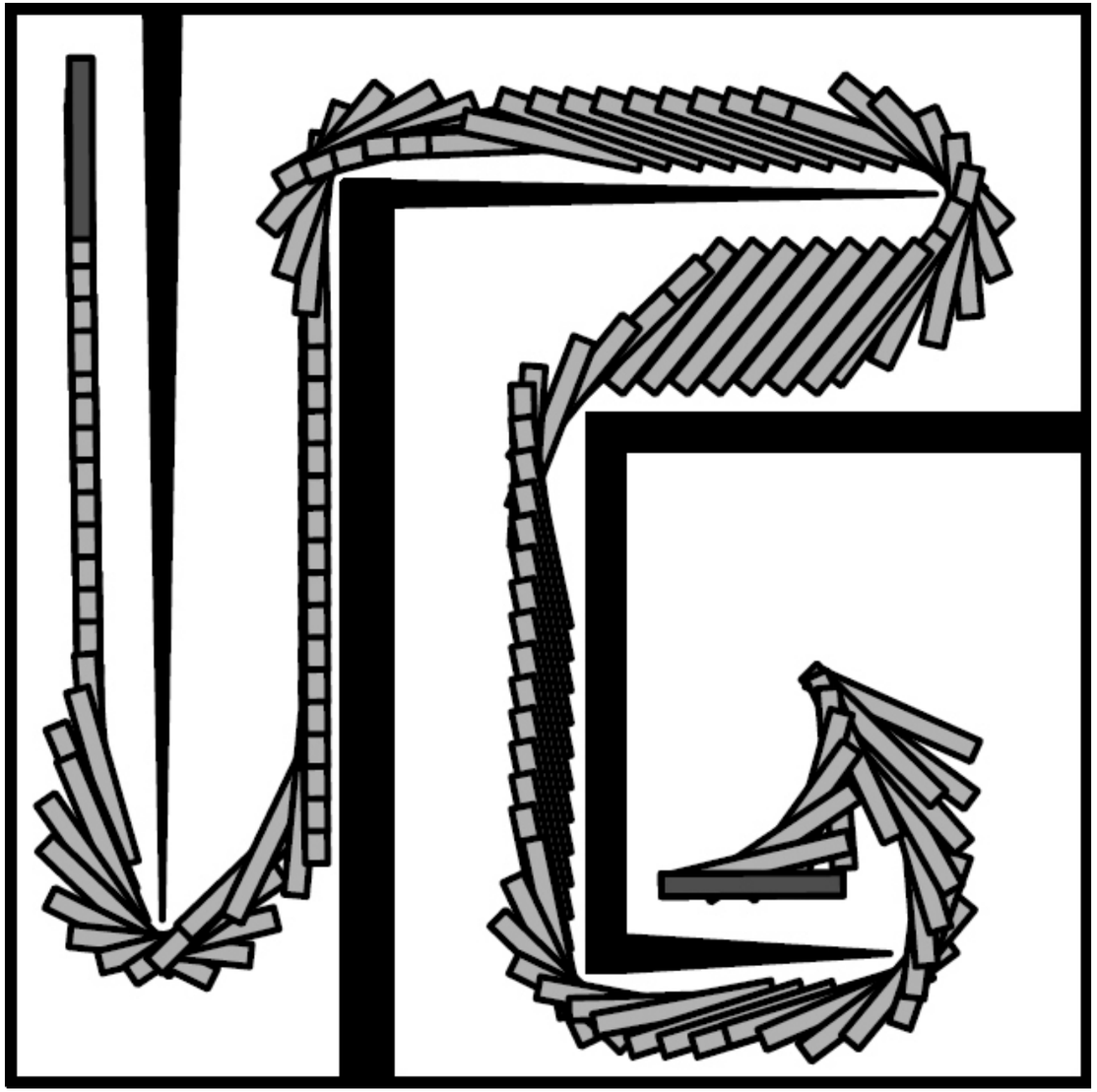
next

variants

- shape of robot **polygonal**
- shape of obstacles **polygonal**
- motion (e.g. rotation) of robot
  translations + rotations!
- motion of obstacles **no motion**
- 2D or 3D **2D**

Problem: Given a robot at position p_{start} in a region with obstacles find a path to the goal position p_{goal} avoiding obstacles.

Polygonal robots with rotations



Polygonal robots with rotations

Configuration space $R(x, y, \phi) \in \mathbb{R}^2 \times [0°, 360°)$

Polygonal robots with rotations

Configuration space $R(x, y, \phi) \in \mathbb{R}^2 \times [0°, 360°)$

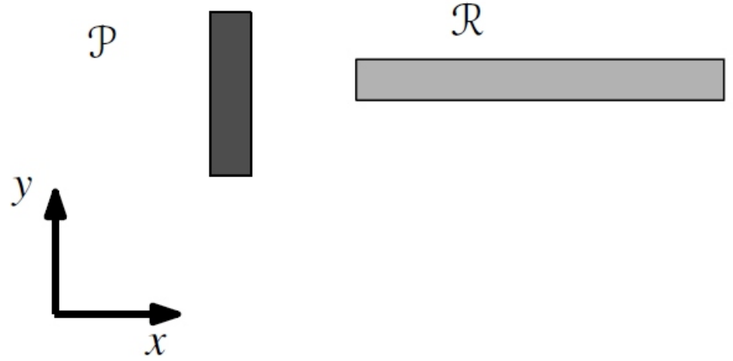
How does $CP = \{(x, y, \phi) : R(x, y, \phi) \cap P \neq \varnothing\}$ look like?

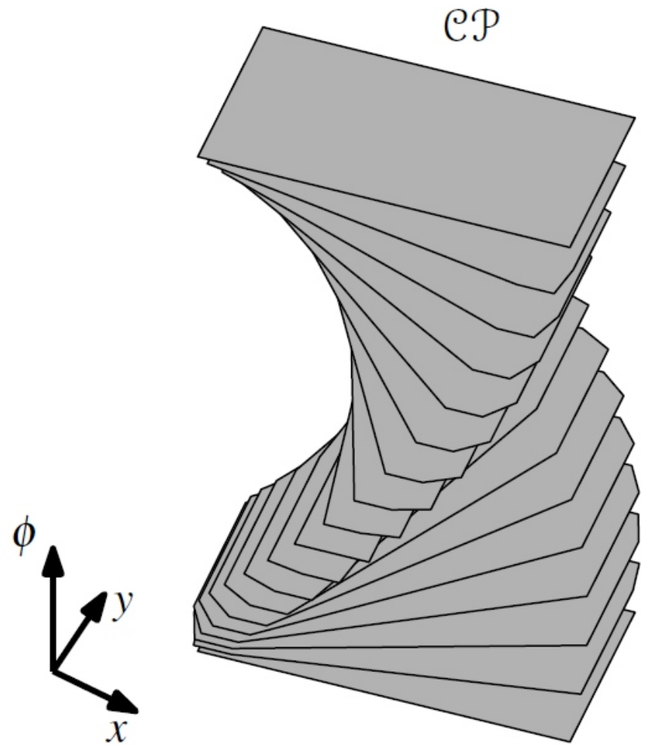
Polygonal robots with rotations

Configuration space $R(x, y, \phi) \in \mathbb{R}^2 \times [0°, 360°)$

How does $CP = \{(x, y, \phi) : R(x, y, \phi) \cap P \neq \varnothing\}$ look like?

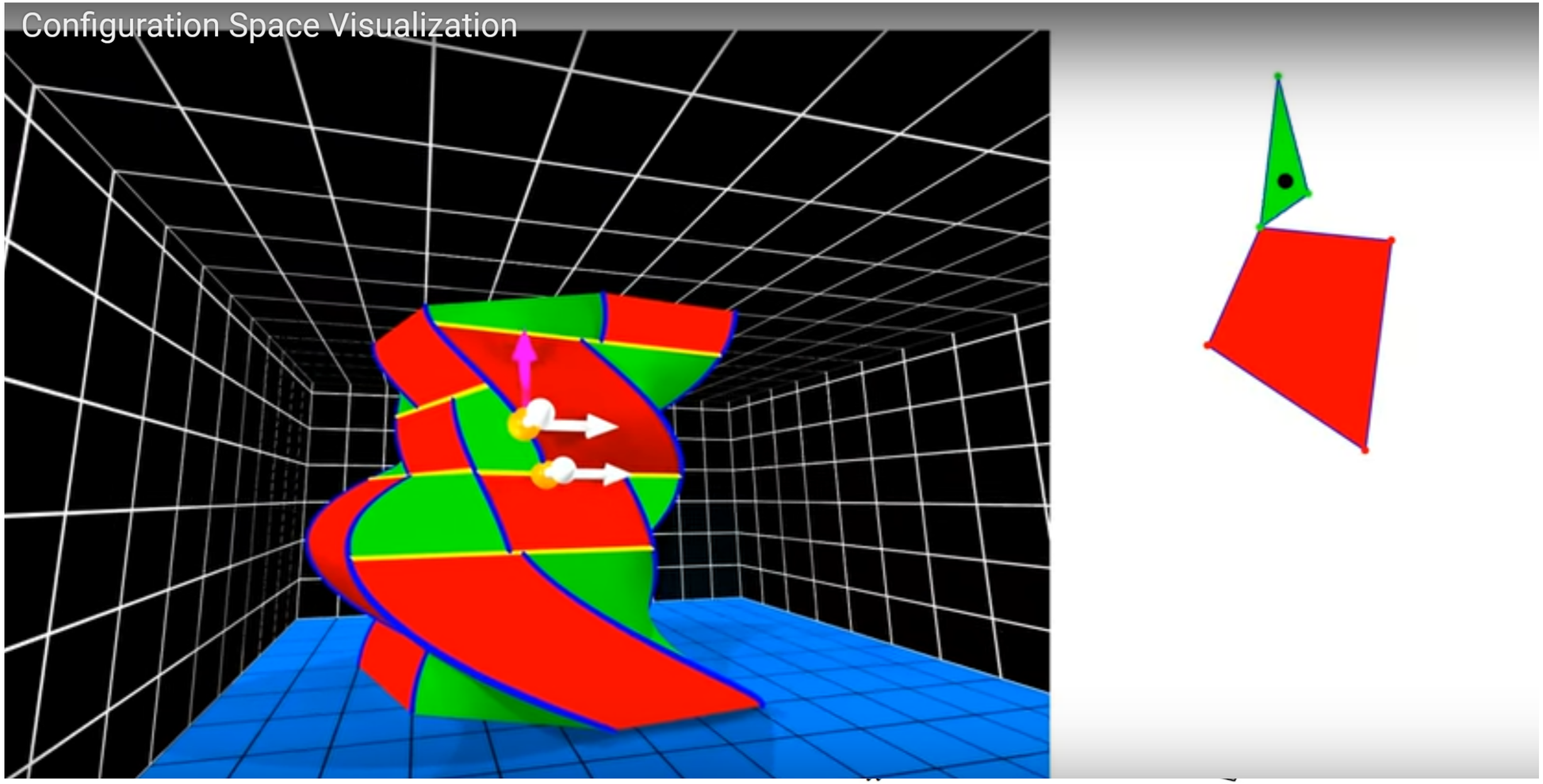
For fixed ϕ it is $P \bigoplus R(0, 0, \phi)$





Polygonal robots with rotations

Configuration space $R(x, y, \phi) \in \mathbb{R}^2 \times [0°, 360°)$



https://youtu.be/SBFwgR4K1Gk

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Compute road map:

1: **for** $i \leftarrow 0$ **to** $z - 1$ **do**
2: compute road map G_i for $\phi_i = i \cdot 360/z$
3: connect G_{i-1} and G_i

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Compute road map:

1: **for** $i \leftarrow 0$ **to** $z - 1$ **do**

2: compute road map G_i for $\phi_i = i \cdot 360/z$

3: connect G_{i-1} and G_i

as always

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Compute road map:

1: **for** $i \leftarrow 0$ **to** $z - 1$ **do**
2: compute road map G_i for $\phi_i = i \cdot 360/z$
3: connect G_{i-1} and G_i

as always

1: compute overlay
2: **for** every joint cell **do**
3: choose $(x, y) \in \Delta \cap \Delta'$
4: add (x, y, ϕ_{i-1}) and (x, y, ϕ_i) to G_{i-1} and G_i, and connect them

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Compute road map:

1: **for** $i \leftarrow 0$ **to** $z - 1$ **do**
2: compute road map G_i for $\phi_i = i \cdot 360/z$
3: connect G_{i-1} and G_i

as always

Is this approach correct?

1: compute overlay
2: **for** every joint cell **do**
3: choose $(x, y) \in \Delta \cap \Delta'$
4: add (x, y, ϕ_{i-1}) and (x, y, ϕ_i) to G_{i-1} and G_i, and connect them

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

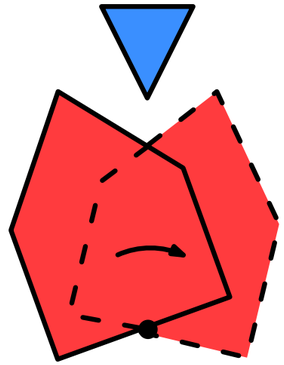
Two types of **errors** occur:

Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

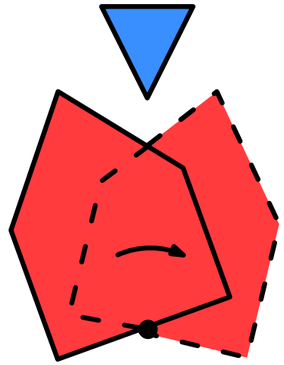
Two types of **errors** occur:

- paths that don’t exist may be allowed



Polygonal robots with rotations

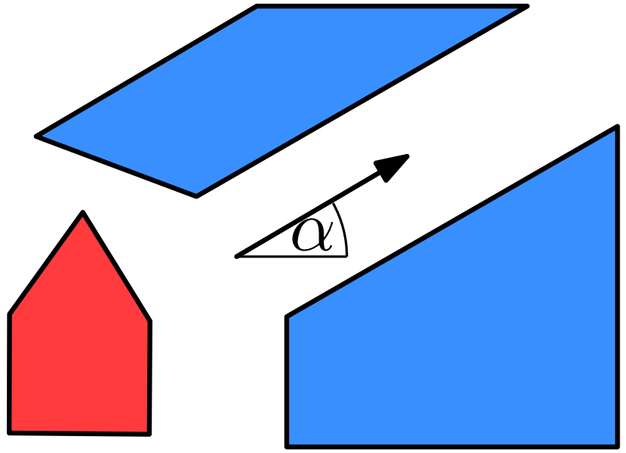
”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)



Two types of **errors** occur:

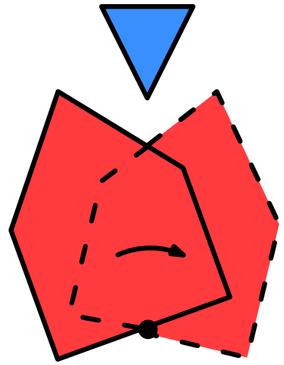
- paths that don’t exist may be allowed

- paths that do exist might not be allowed



Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

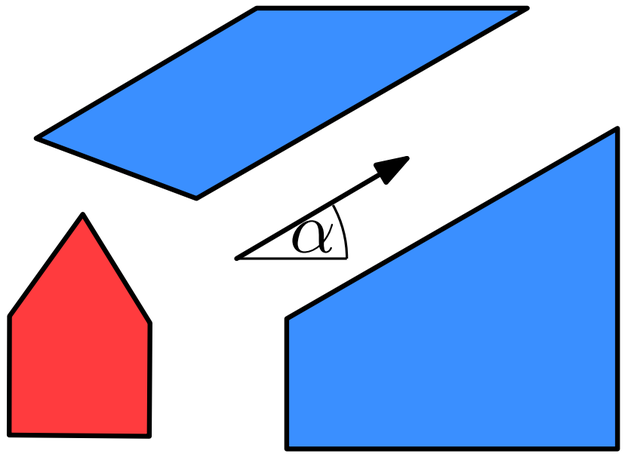


Two types of **errors** occur:

- paths that don’t exist may be allowed

  How can we avoid this?

- paths that do exist might not be allowed



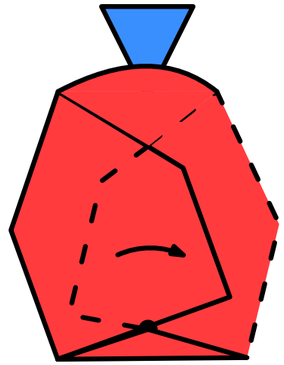
Polygonal robots with rotations

”Simple“ approach: take a finite number of horizontal slices in the configuration space ($\Rightarrow$ discretize rotations)

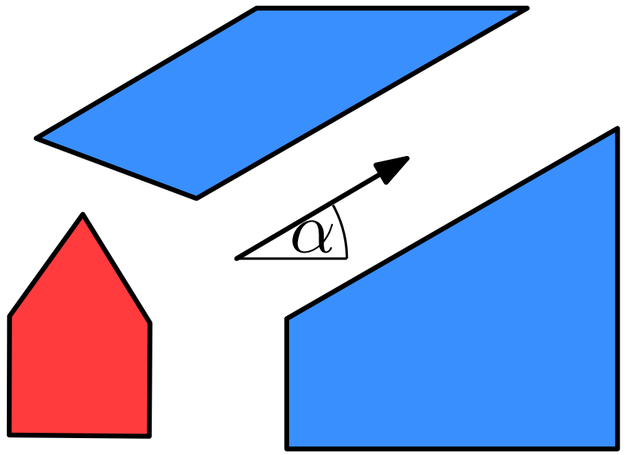
Two types of **errors** occur:

- paths that don’t exist may be allowed

  How can we avoid this? Enlarge robot

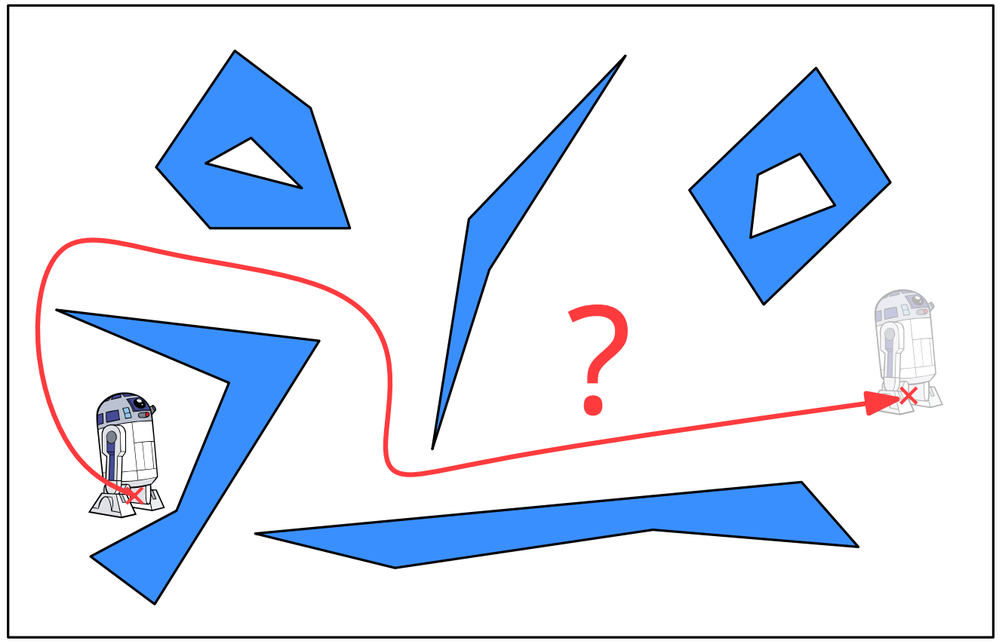


- paths that do exist might not be allowed



“Middle” path

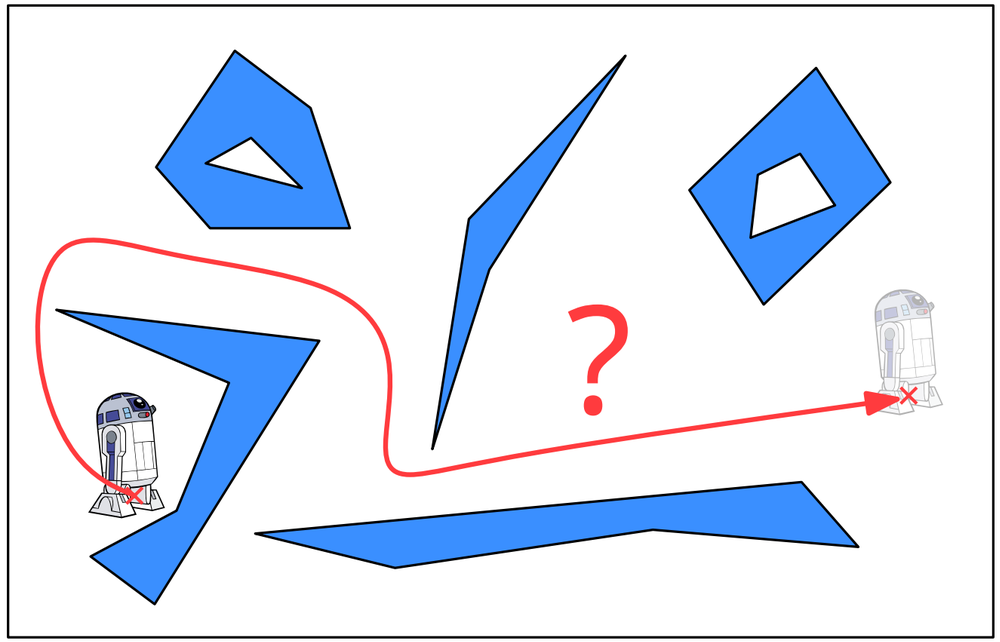
How can we find a path with distance as large as possible to obstacles?



“Middle” path

How can we find a path with distance as large as possible to obstacles?

If robot and obstacles are disks?



with a

A: vertical decomposition

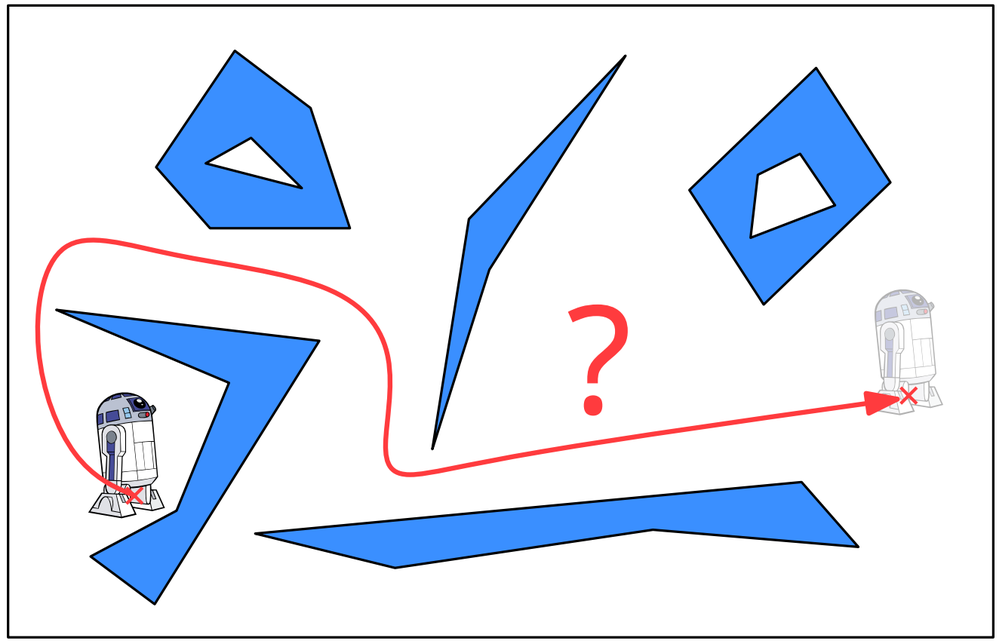
B: Voronoi diagram

C: line arrangement

“Middle” path

How can we find a path with distance as large as possible to obstacles?

If robot and obstacles are disks?



with a

A: vertical decomposition

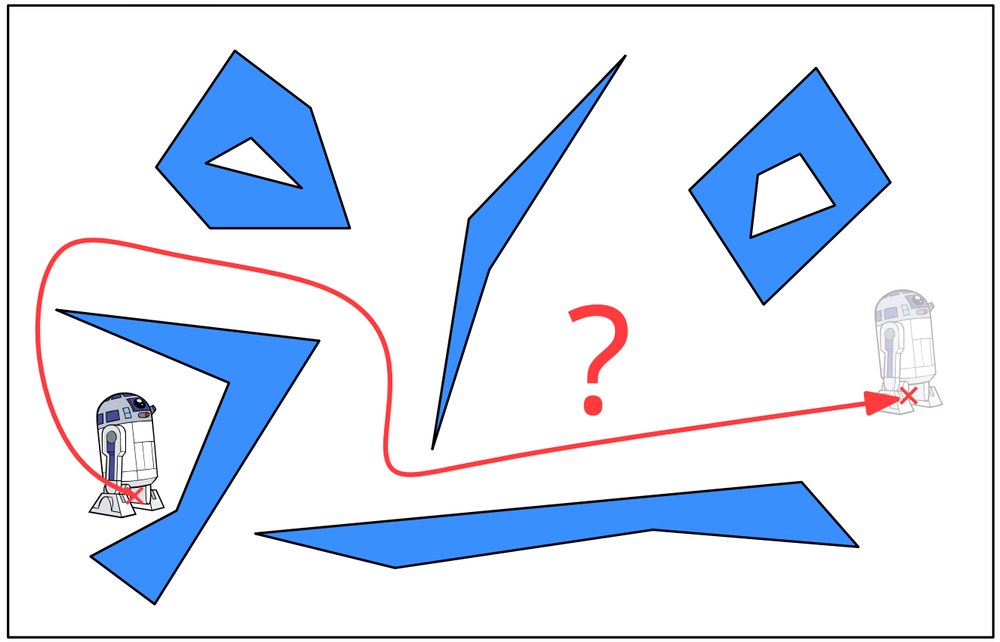
B: Voronoi diagram

C: line arrangement

“Middle” path

How can we find a path with distance as large as possible to obstacles?

If robot and obstacles are disks?



with a

A: vertical decomposition

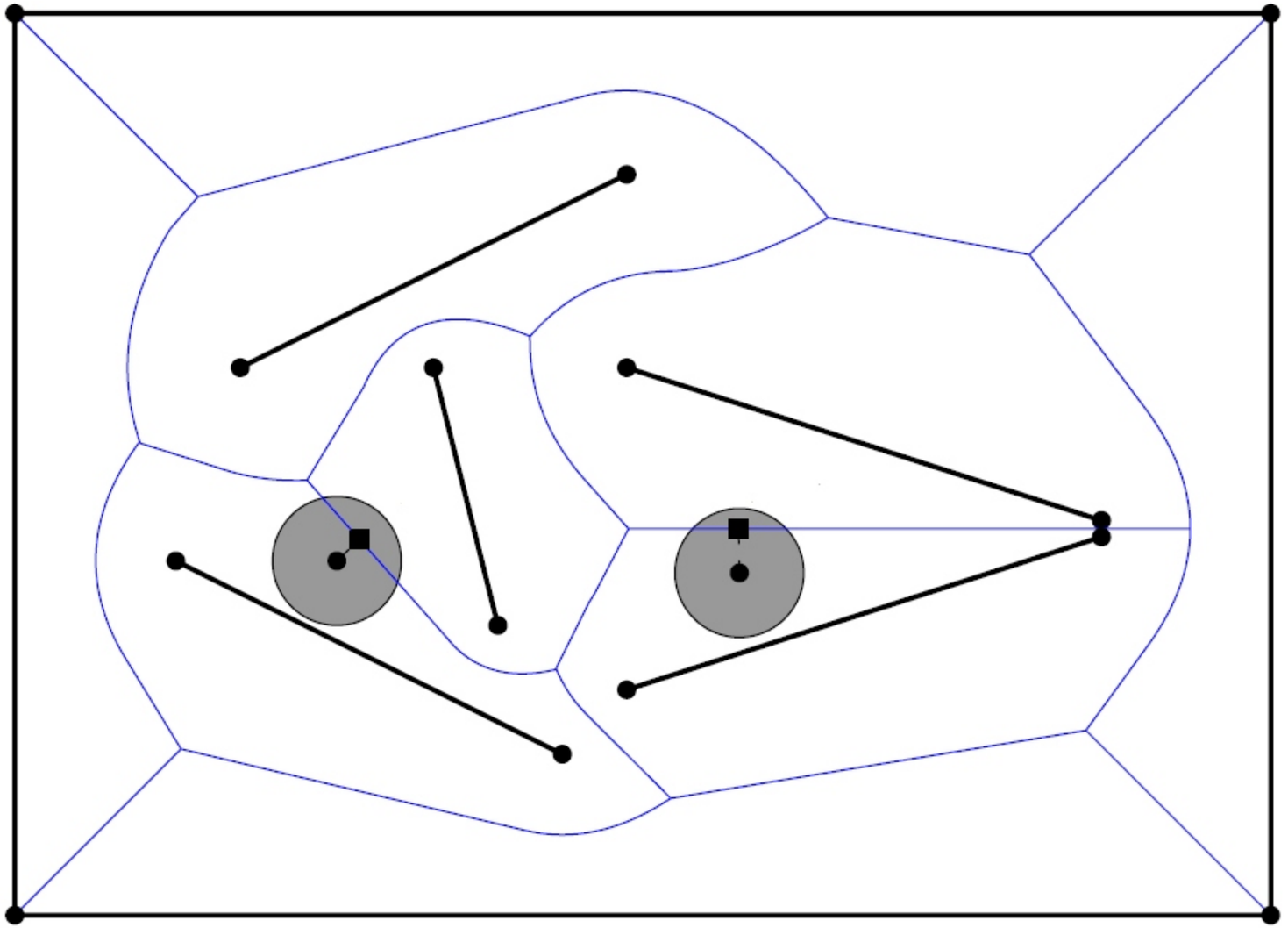
B: Voronoi diagram

C: line arrangement

And if obstacles are polygonal?

“Middle” path

Approach: Use *Voronoi diagram* of edges of the obstacles as graph (see book).

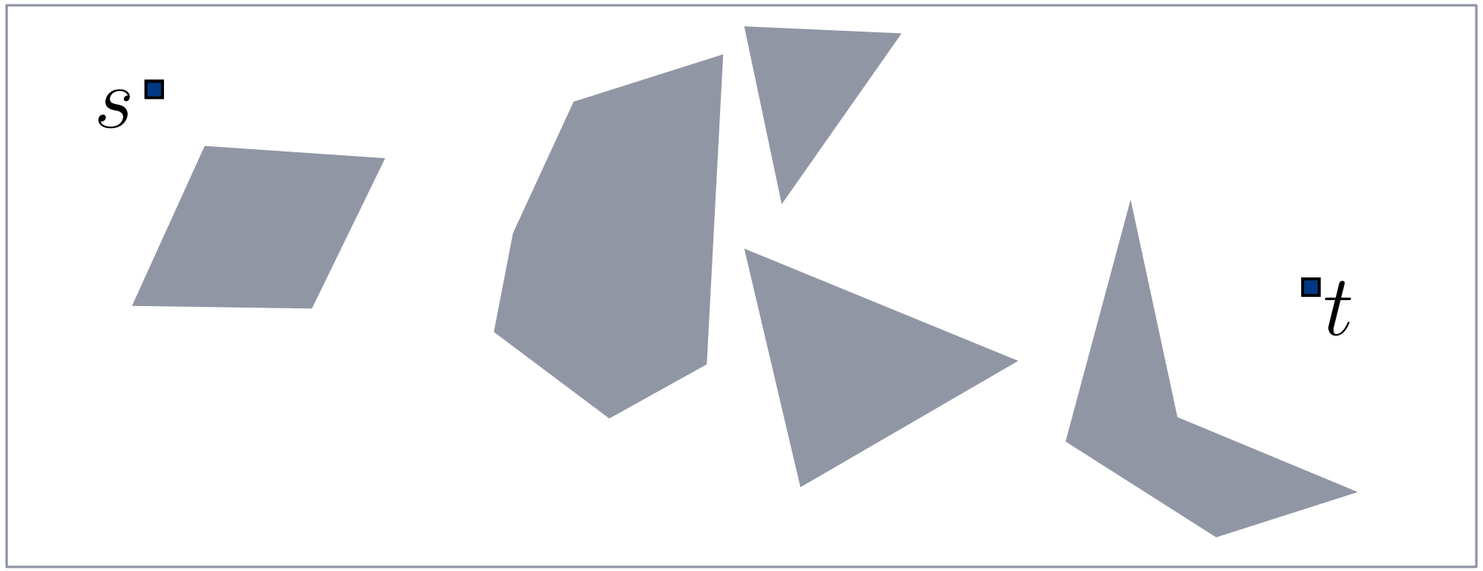


Robot Motion Planning

visibility graphs

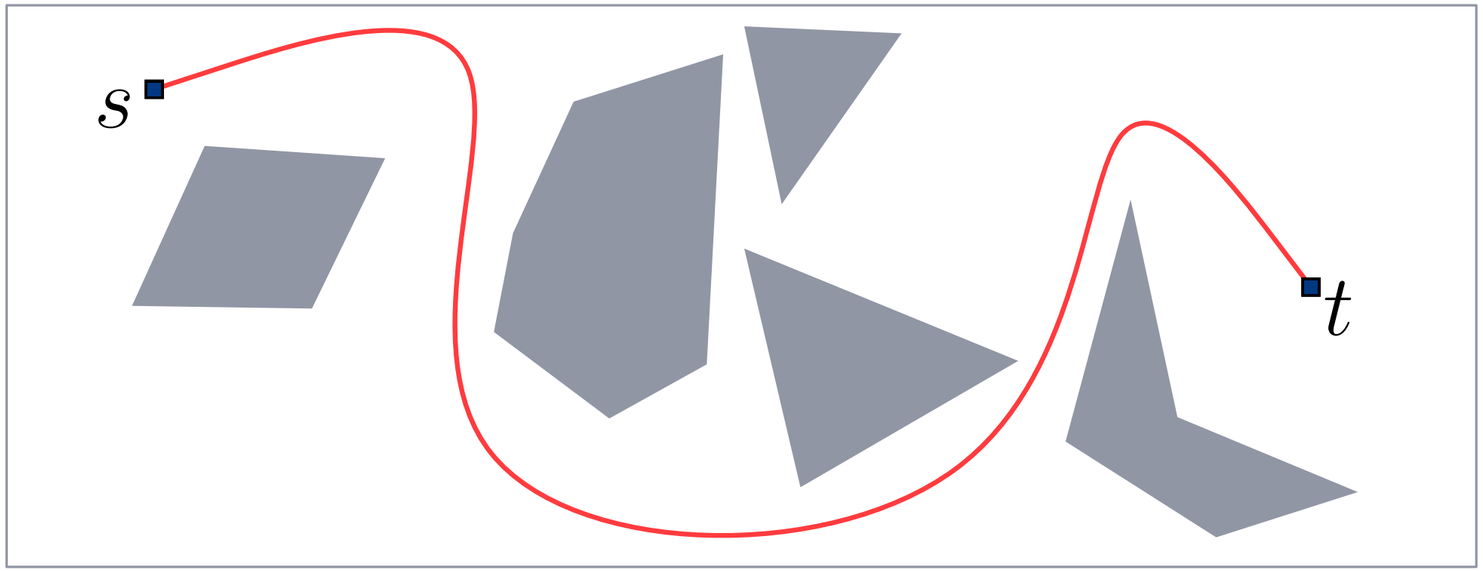
Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



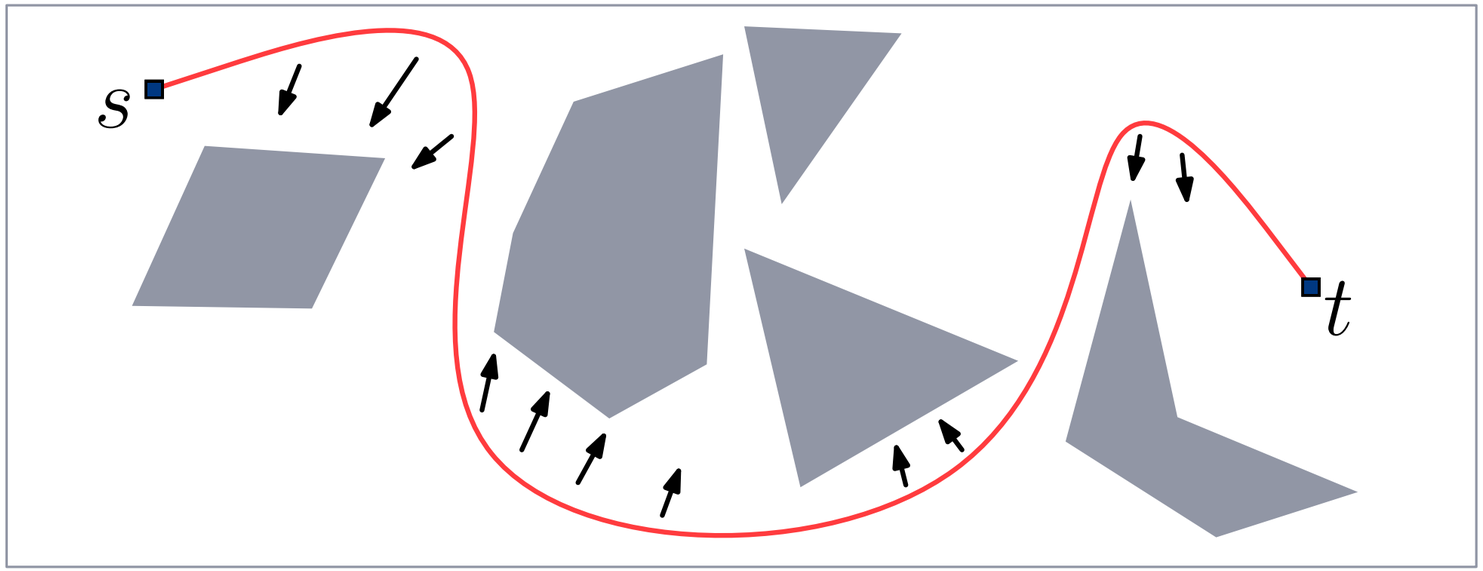
Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



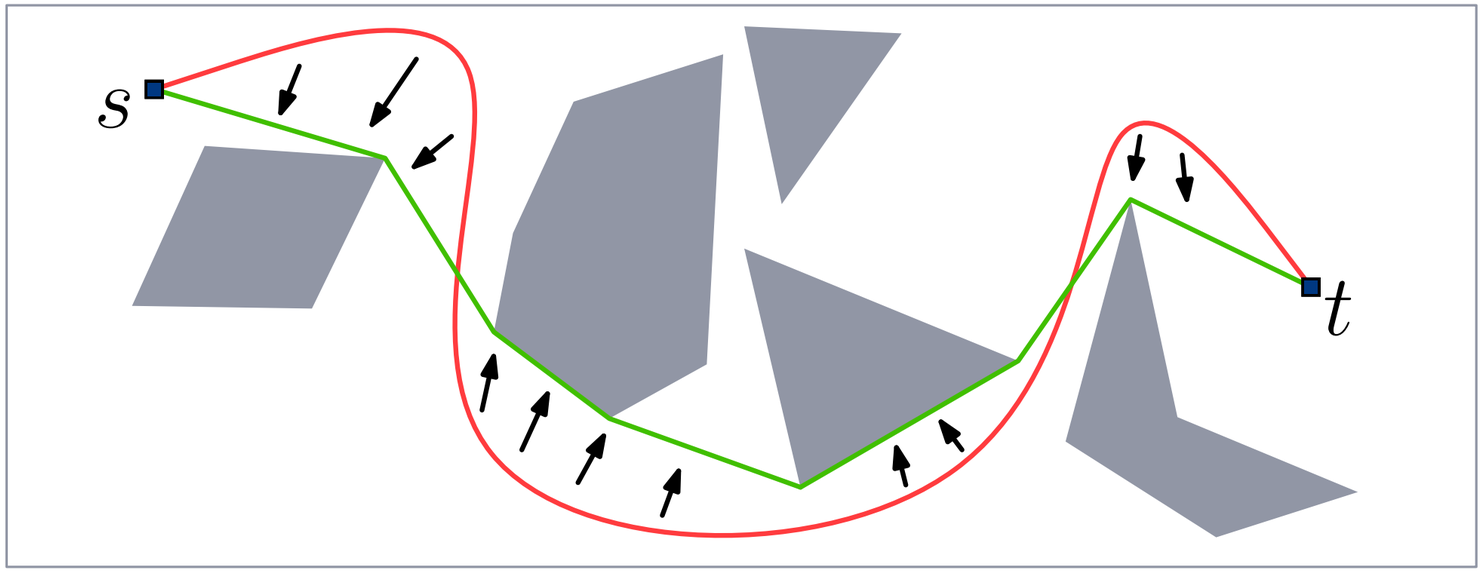
Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



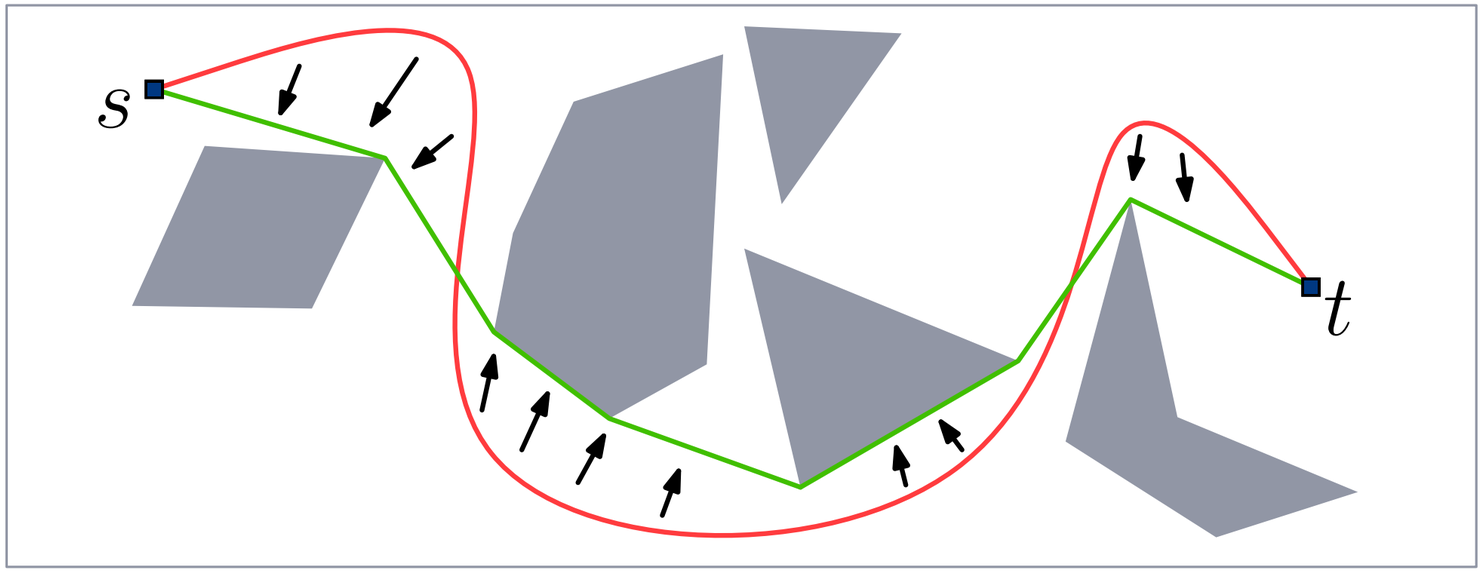
Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



Shortest paths in polygonal domains

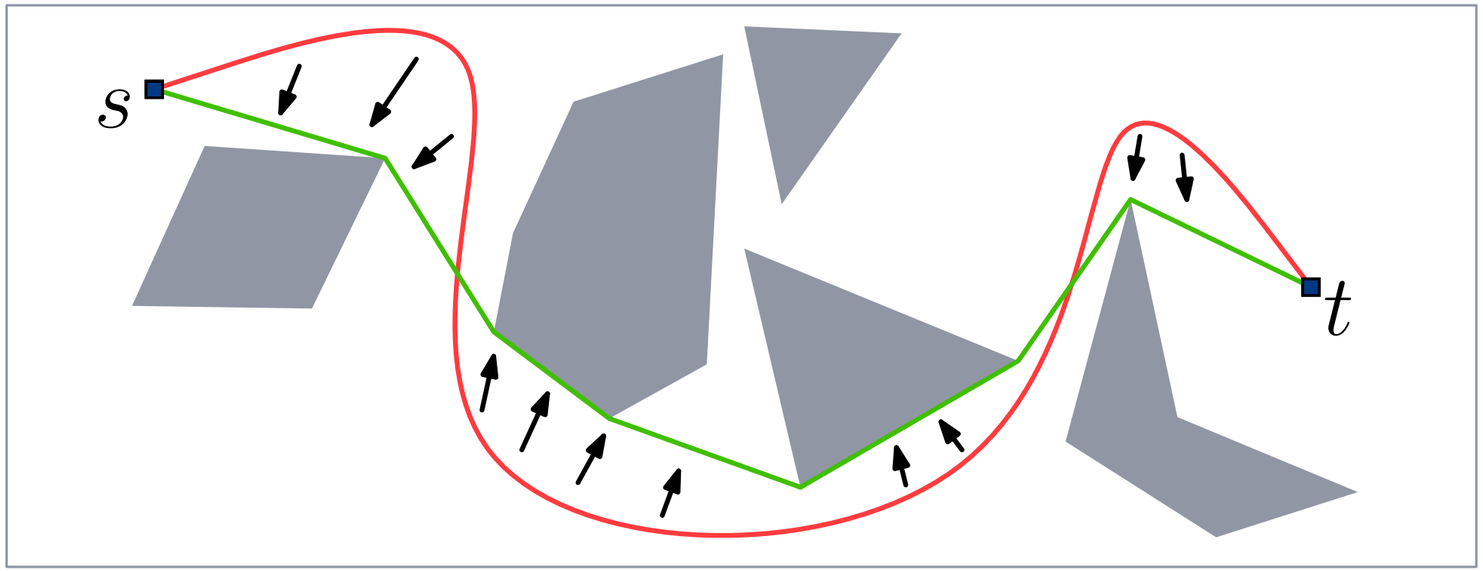
Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



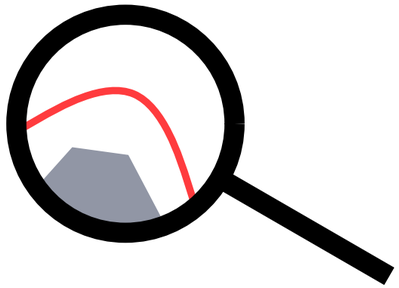
Proof sketch:

Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

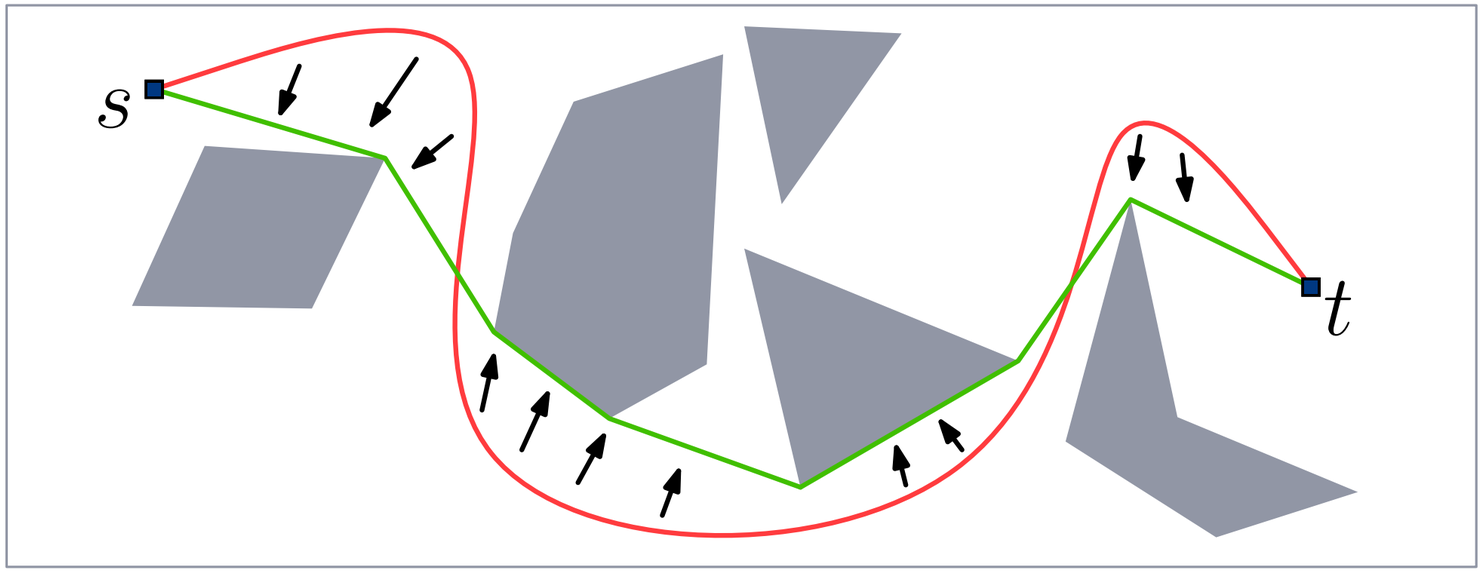


Proof sketch:

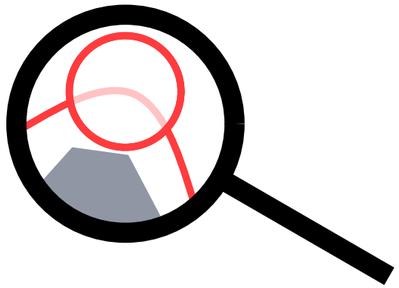


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

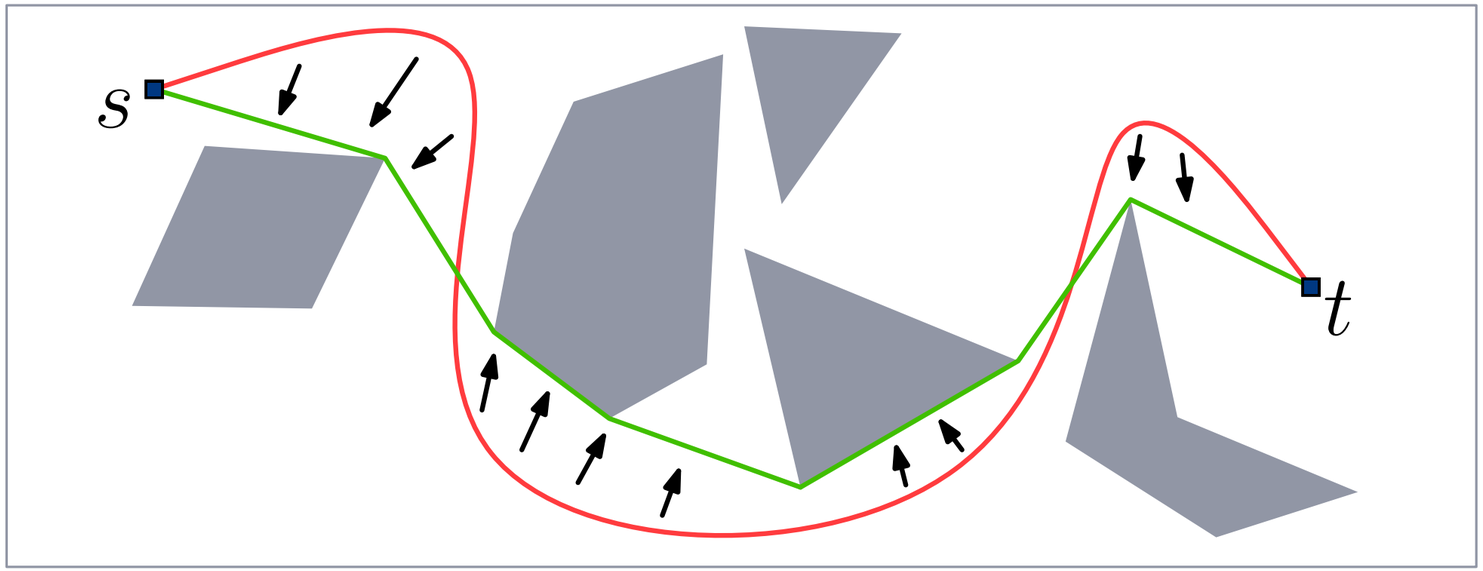


Proof sketch:

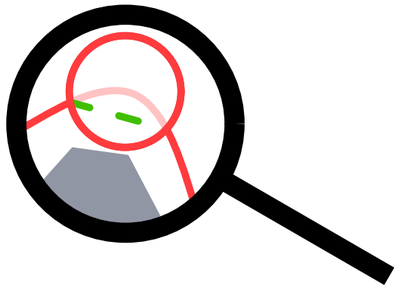


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

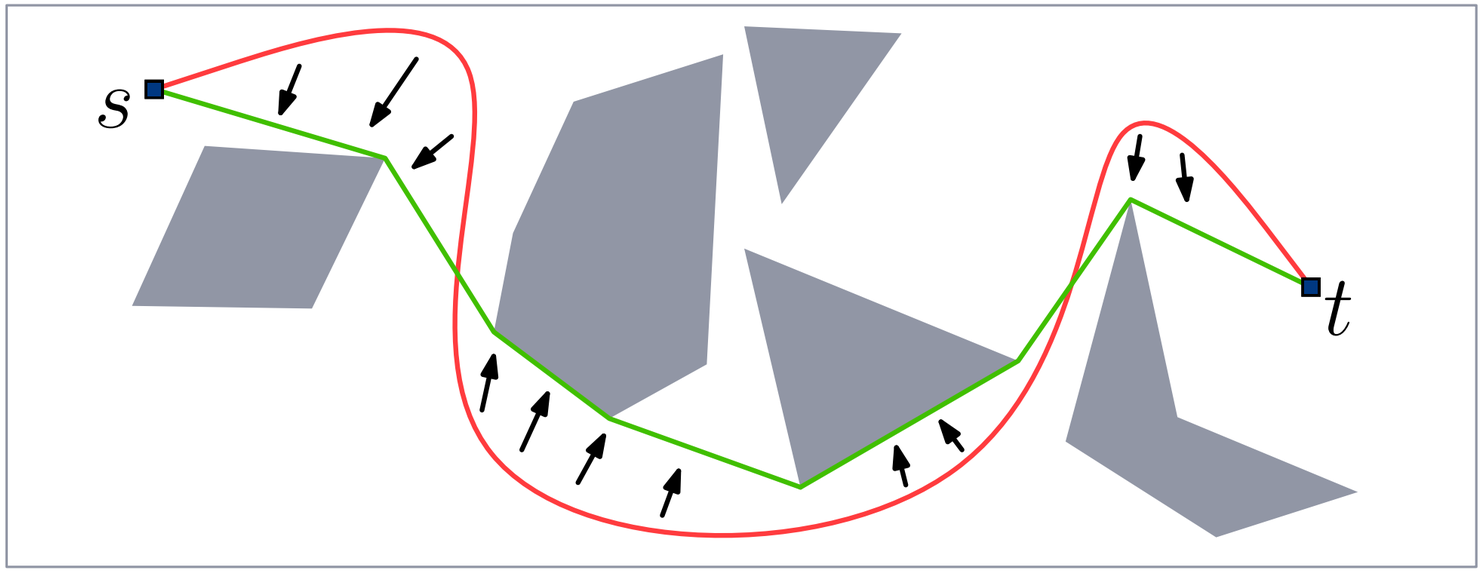


Proof sketch:

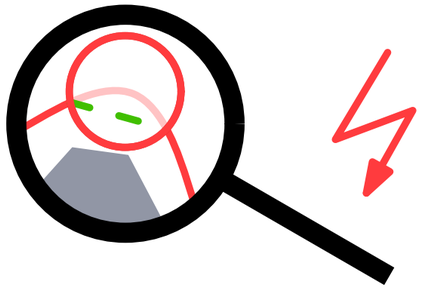


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

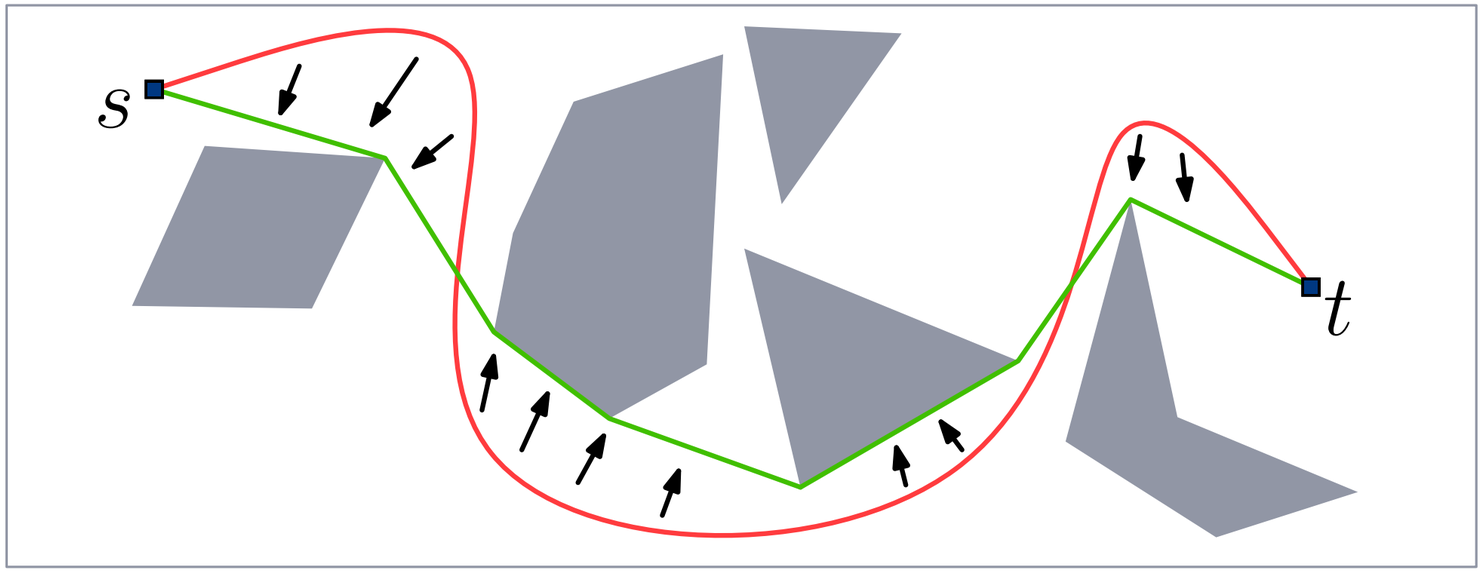


Proof sketch:

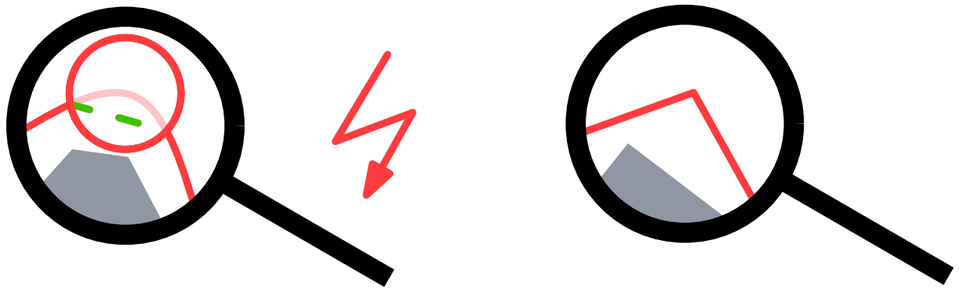


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

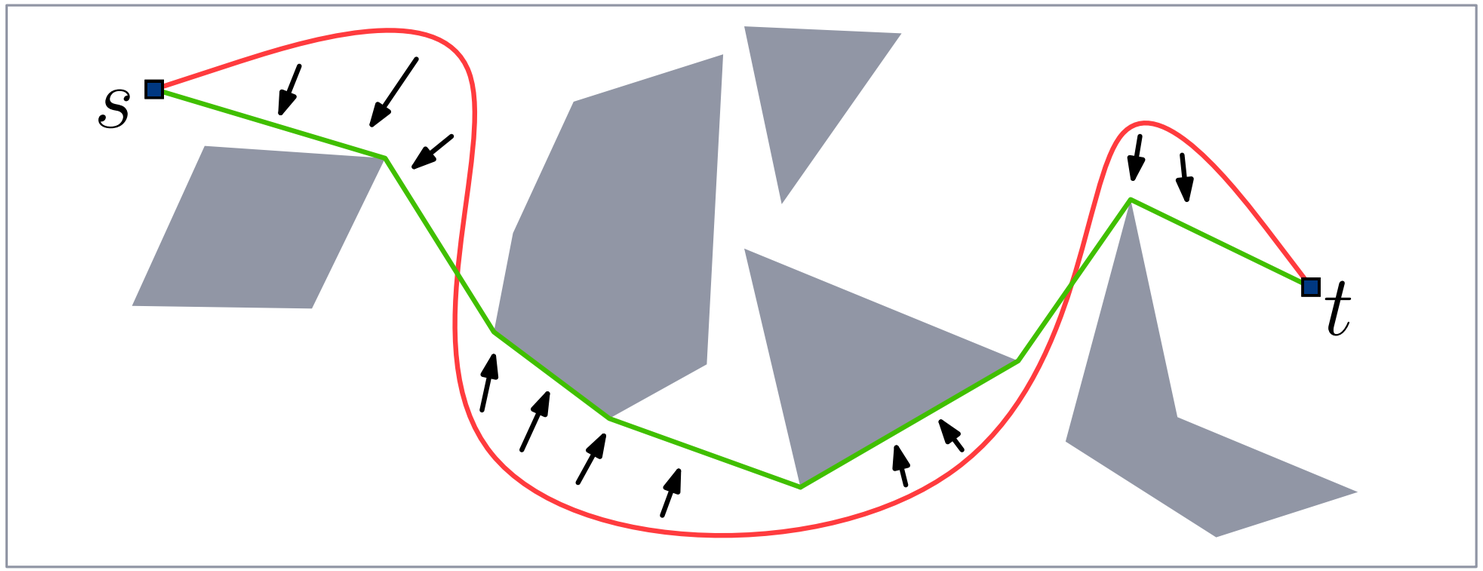


Proof sketch:

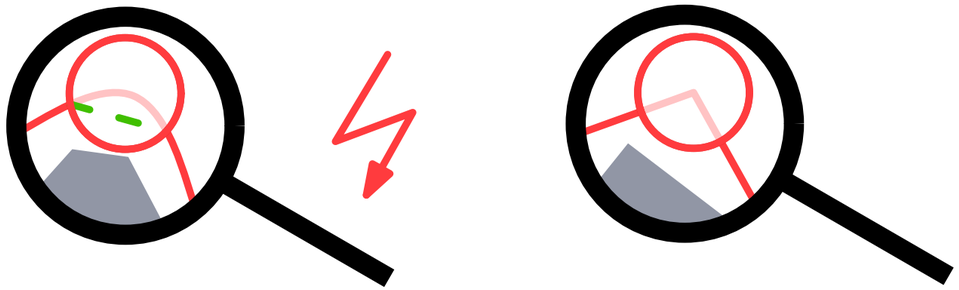


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

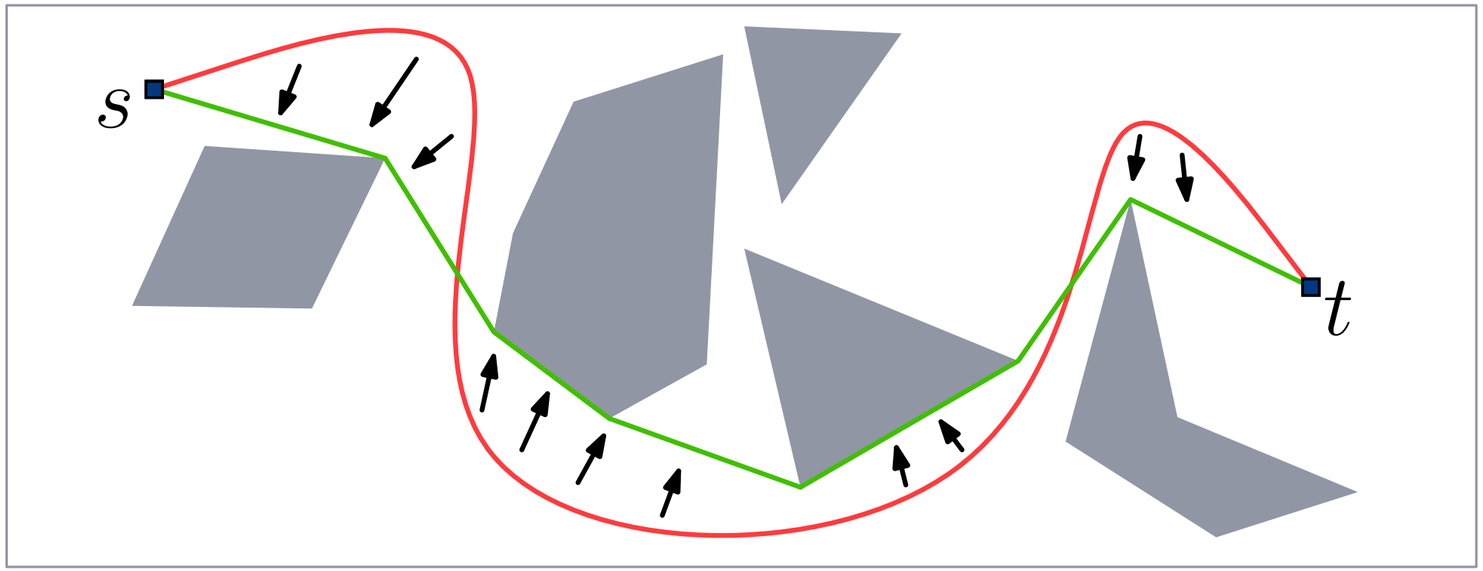


Proof sketch:

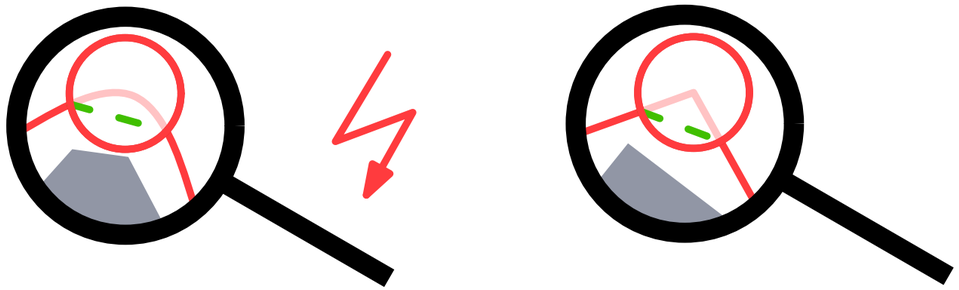


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).

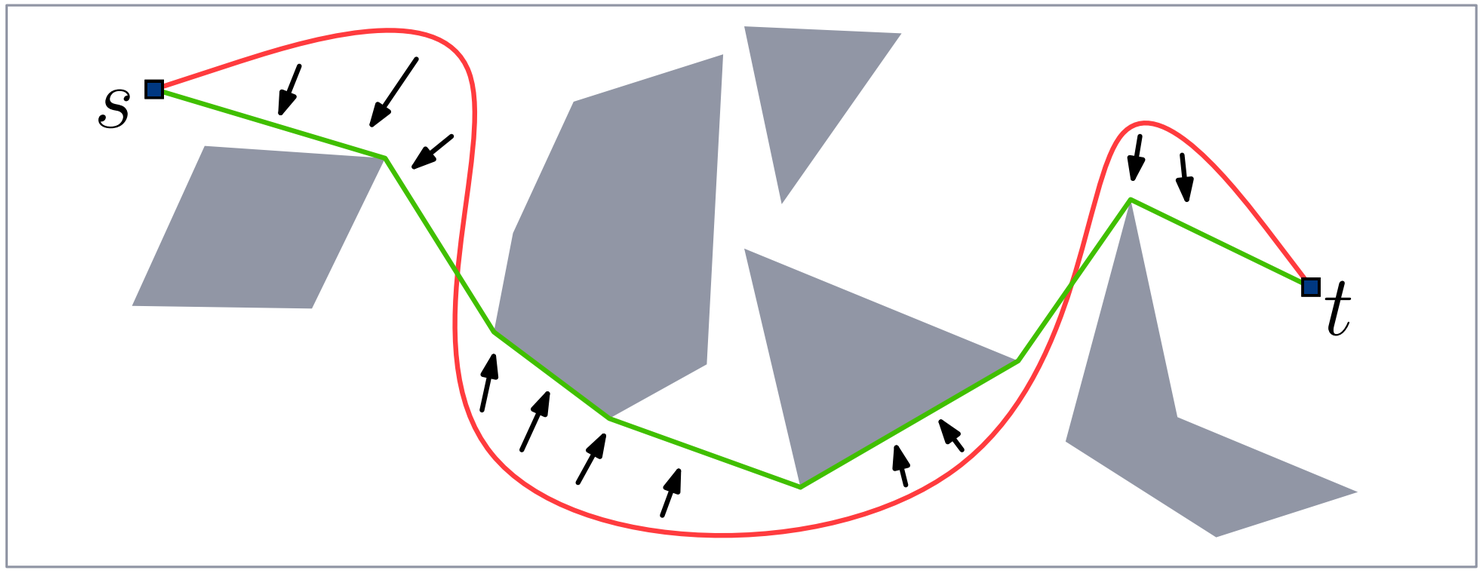


Proof sketch:

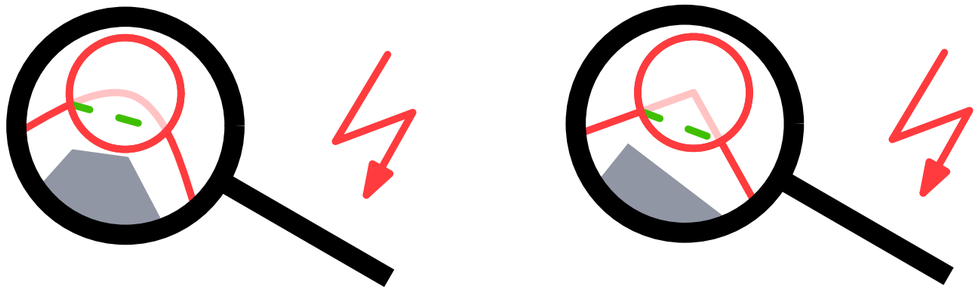


Shortest paths in polygonal domains

Lemma 1: For a set S of disjoint polygons in $\mathbb{R}^2$ and two points s and t outside of S, every shortest st-path in $\mathbb{R}^2 \setminus \bigcup S$ is a polygonal curve with vertices of polygons in S (and s, t).



Proof sketch:

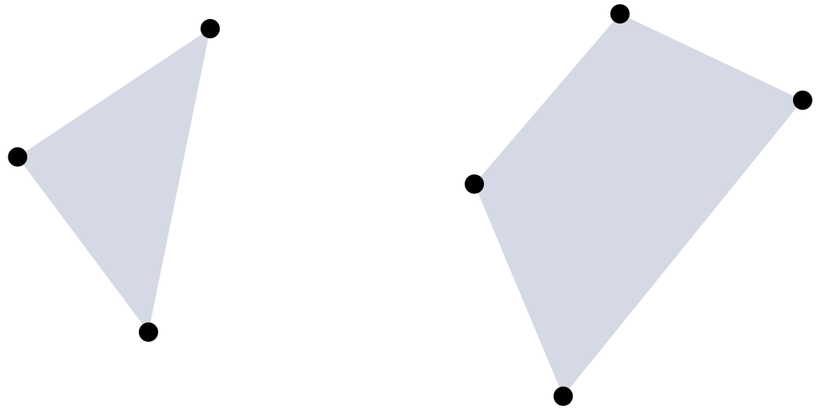


Visibility graph

Given a set S of disjoint, open polygons with vertex set $V(S)$.

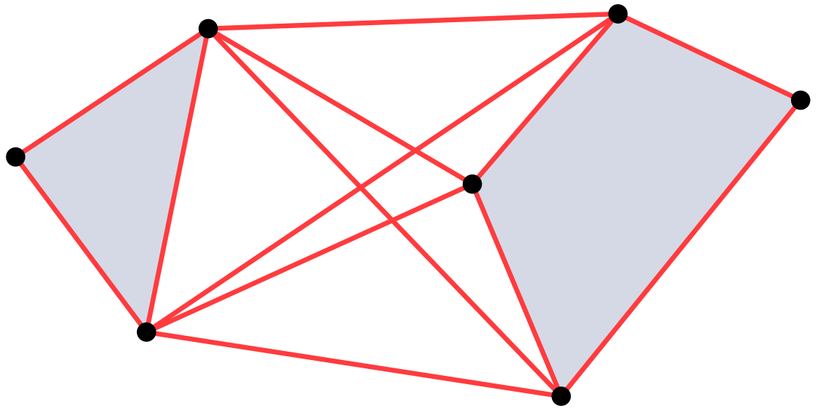
Visibility graph

Given a set S of disjoint, open polygons with vertex set $V(S)$.



Visibility graph

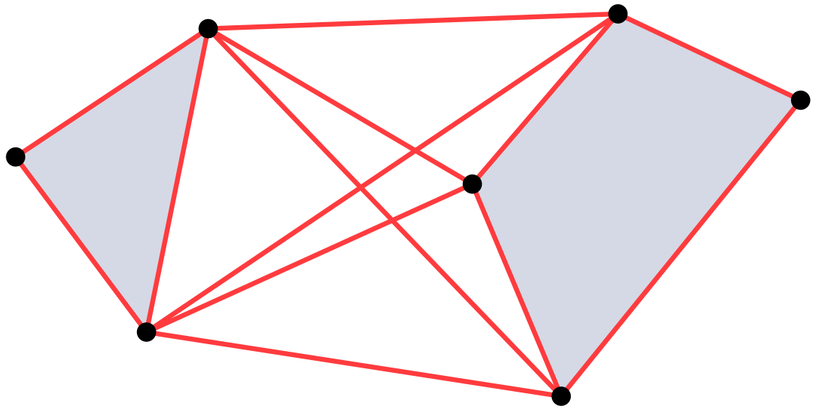
Given a set S of disjoint, open polygons with vertex set $V(S)$.



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

Visibility graph

Given a set S of disjoint, open polygons with vertex set $V(S)$.

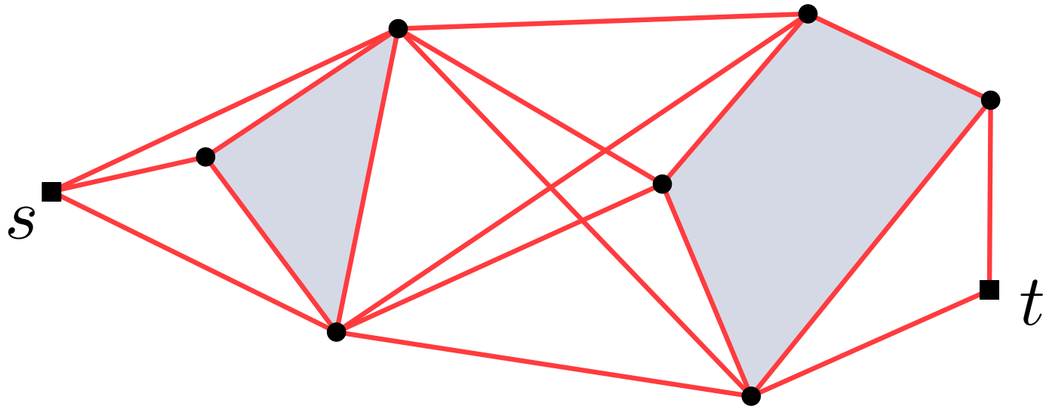


Definition: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the visibility graph of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S) \text{ and } u \text{ sees } v\}$ and $w(uv) = |uv|$.

Here u sees $v \Leftrightarrow \overline{uv} \cap \bigcup S = \varnothing$

Visibility graph

Given a set S of disjoint, open polygons with vertex set $V(S)$.



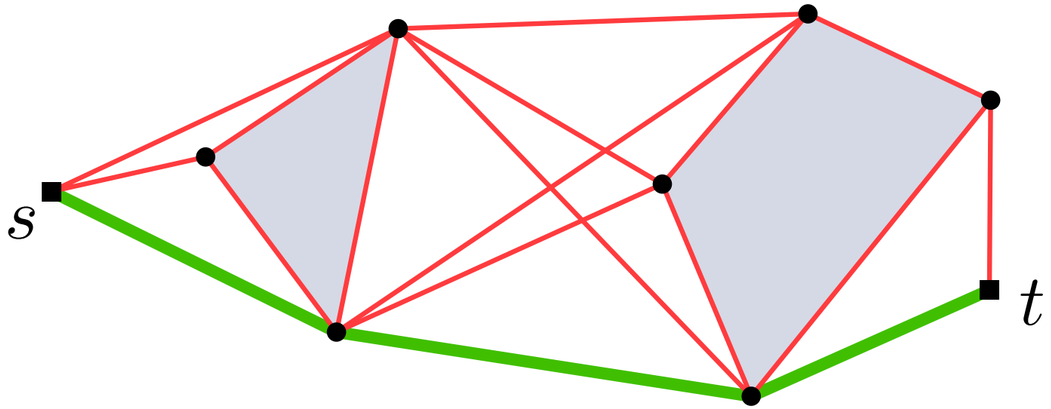
Definition: Then $G_{\text{vis}}(S) = (V(S), E_{\text{vis}}(S))$ is the visibility graph of S with $E_{\text{vis}}(S) = \{uv \mid u, v \in V(S) \text{ and } u \text{ sees } v\}$ and $w(uv) = |uv|$.

Here u sees $v \Leftrightarrow \overline{uv} \cap \bigcup S = \varnothing$

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

Visibility graph

Given a set S of disjoint, open polygons with vertex set $V(S)$.



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

Here u sees $v \Leftrightarrow \overline{uv} \cap \bigcup S = \varnothing$

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

Lemma 1: The shortest st-path that avoids the obstacles in S, is the shortest path in $G_{\text{vis}}(S^*)$.

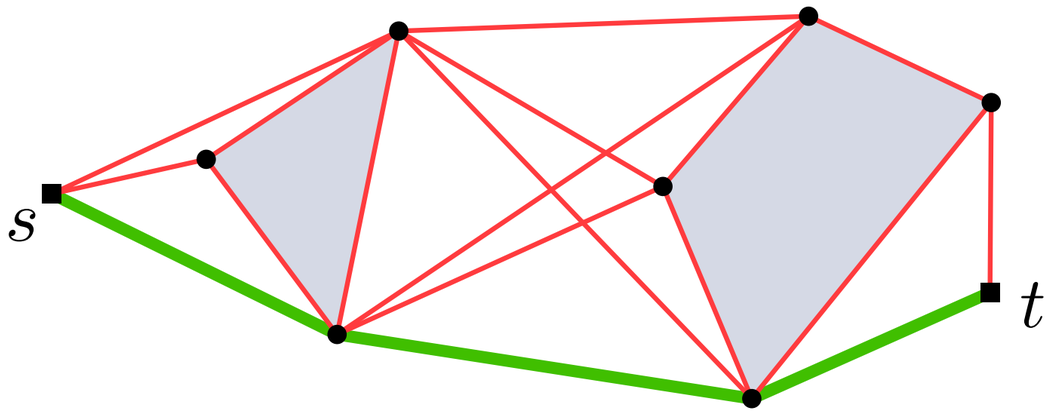
Shortest path computation

ShortestPath(S, s, t)

Input: set of 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$(S \cup \{s, t\})$
2: **for all** $uv \in E_{\text{vis}}$ **do**
3: $\quad w(uv) \leftarrow |uv|$
4: $\quad$ **return** Dijkstra$(G_{\text{vis}}, w, s, t)$



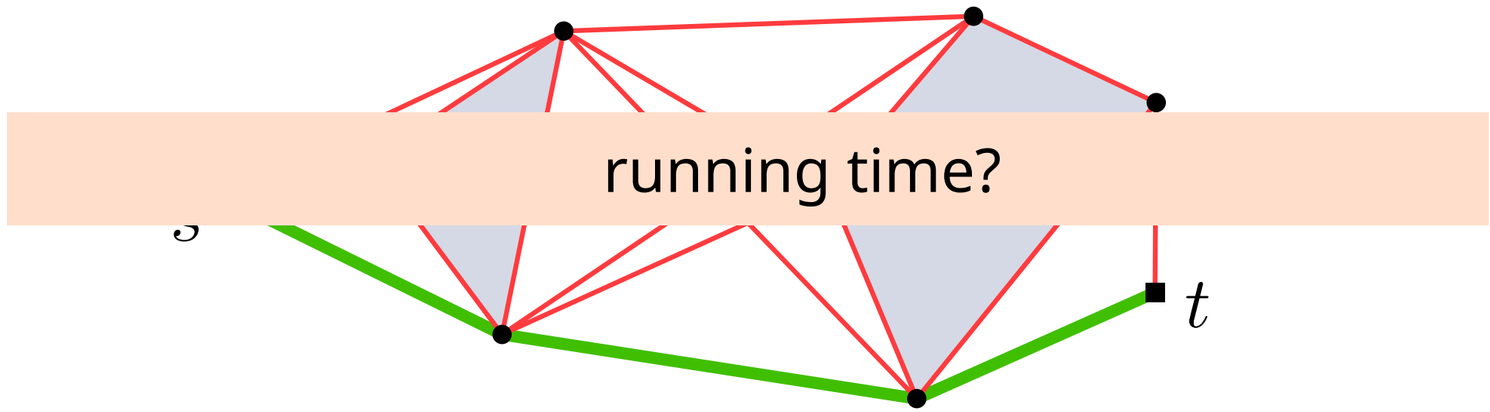
Shortest path computation

SHORTESTPATH(S, s, t)

Input: set of 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$(S \cup \{s, t\})$
2: **for all** $uv \in E_{\text{vis}}$ **do**
3: $\quad w(uv) \leftarrow |uv|$
4: $\quad$ **return** DIJKSTRA$(G_{\text{vis}}, w, s, t)$



Shortest path computation

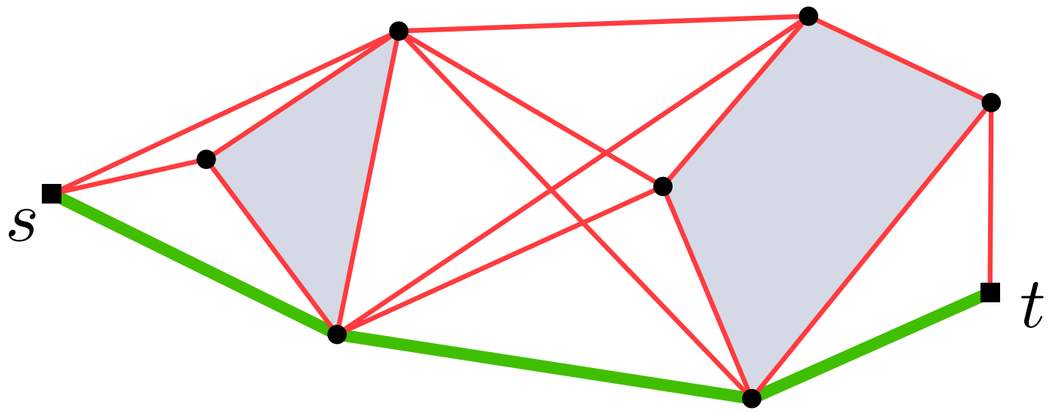
$n = |V(S)|, m = |E_{\text{vis}}(S)|$

SHORTESTPATH(S, s, t)

Input: set of 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$(S \cup \{s, t\})$
2: **for all** $uv \in E_{\text{vis}}$ **do**
3: $\quad w(uv) \leftarrow |uv|$
4: $\quad$ **return** DIJKSTRA$(G_{\text{vis}}, w, s, t)$



Shortest path computation

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

SHORTESTPATH(S, s, t)

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

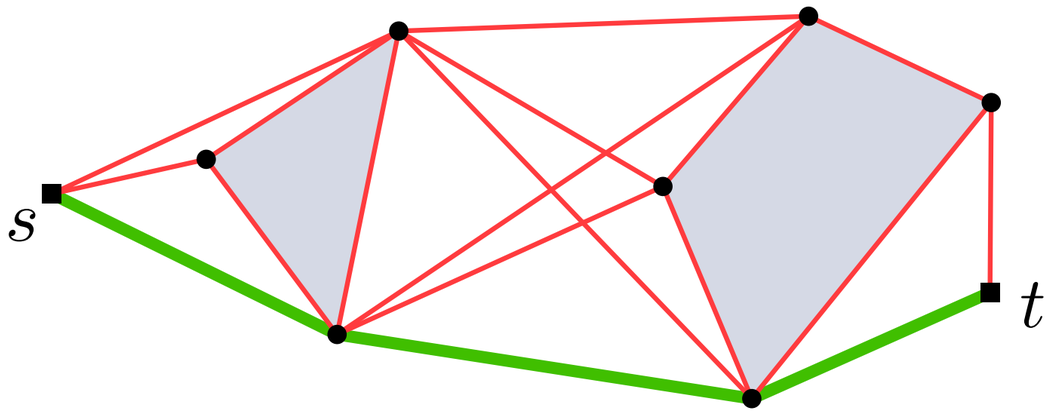
Output: shortest collision-free st-path in S

1: $G_{\text{vis}} \leftarrow$ VISIBILITYGRAPH$(S \cup \{s, t\})$ $\quad O(m)$

2: **for all** $uv \in E_{\text{vis}}$ **do**

3: $\quad w(uv) \leftarrow |uv|$

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



Shortest path computation

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

SHORTESTPATH(S, s, t)

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

Output: shortest collision-free st-path in S

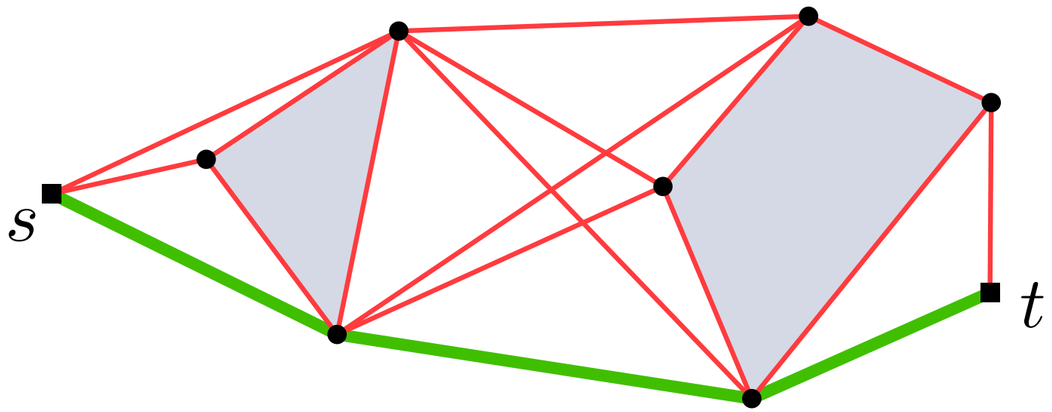
1: $G_{\text{vis}} \leftarrow$ VISIBILITYGRAPH$(S \cup \{s, t\})$ — $O(n^3)$

2: **for all** $uv \in E_{\text{vis}}$ **do**

3: $\quad w(uv) \leftarrow |uv|$ — $O(m)$

4: $\quad$ **return** DIJKSTRA$(G_{\text{vis}}, w, s, t)$ — $O(n \log n + m)$

naive $O(n^3)$
faster?



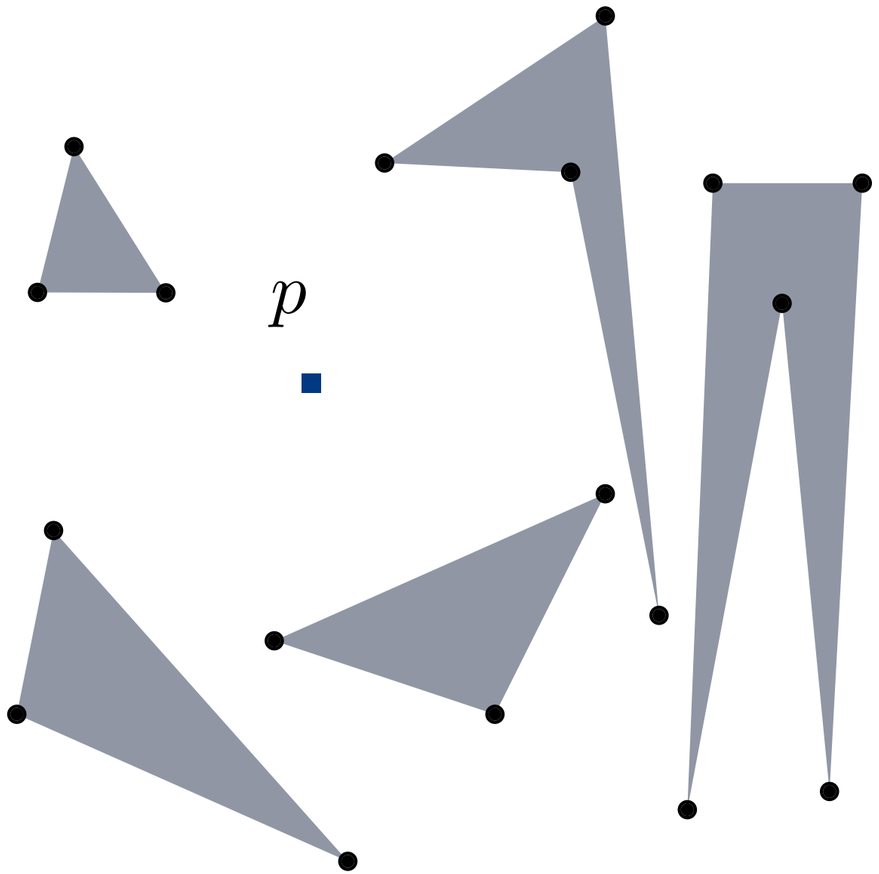
Shortest path computation

How fast can we compute all vertices visible from a given vertex p?

A: $O(n \log n)$

B: $O(n^2)$

C: $O(n^2 \log n)$



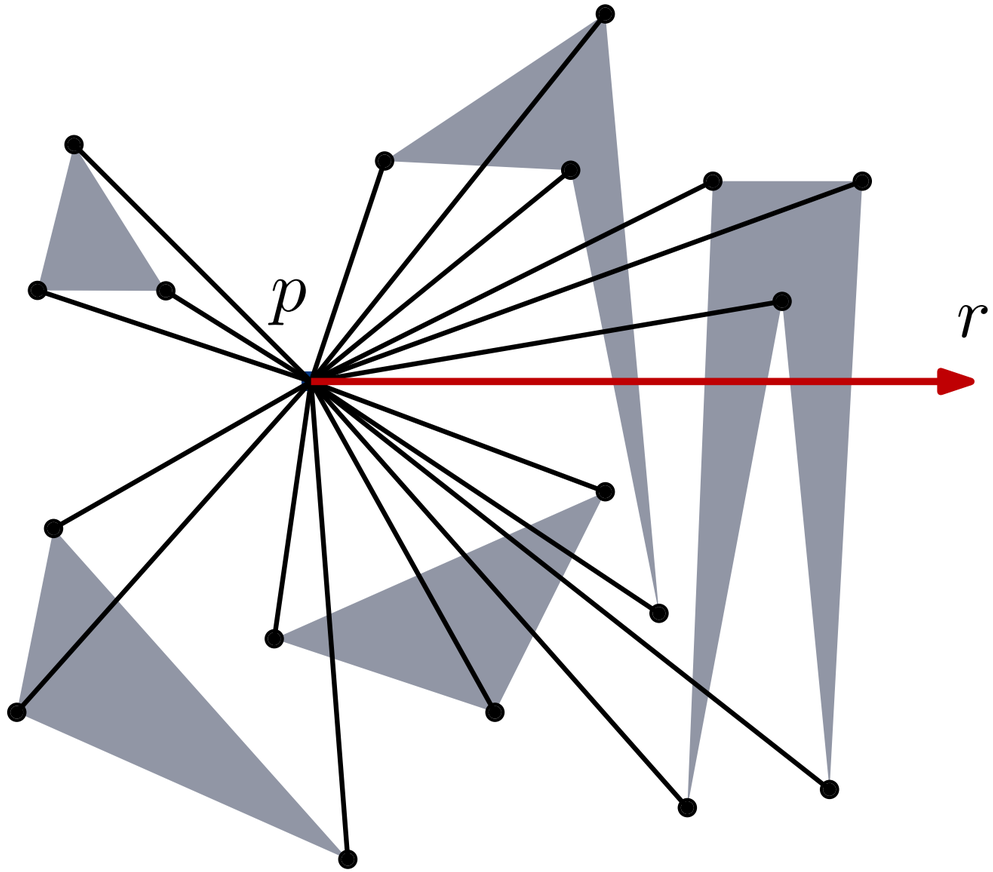
Shortest path computation

How fast can we compute all vertices visible from a given vertex p?

A: $O(n \log n)$

B: $O(n^2)$

C: $O(n^2 \log n)$



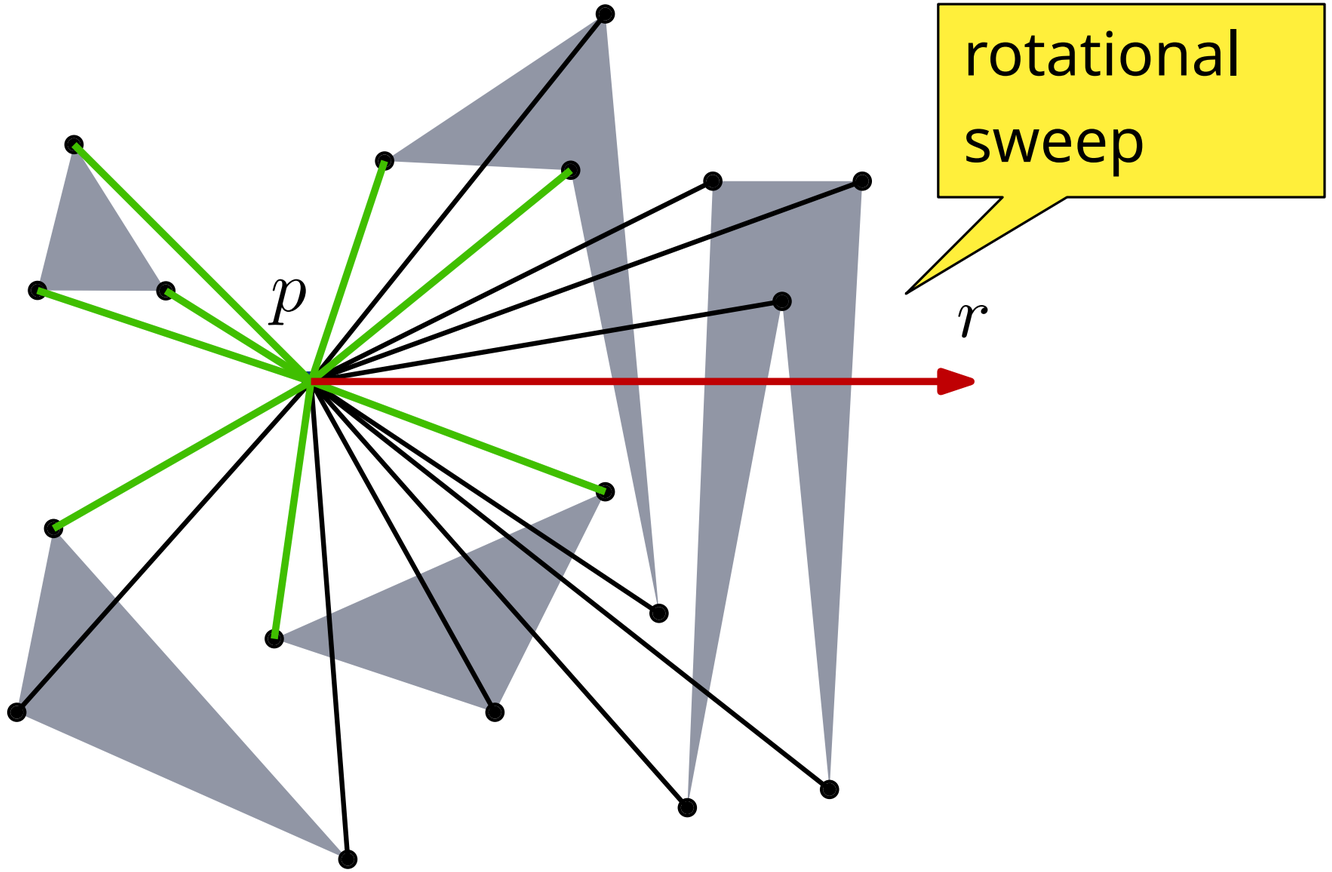
Shortest path computation

How fast can we compute all vertices visible from a given vertex p?

A: $O(n \log n)$

B: $O(n^2)$

C: $O(n^2 \log n)$



Shortest path computation

$\textsc{ShortestPath}(S, s, t)$

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

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

Output: shortest collision-free st-path in S

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

2: **for all** $uv \in E_{\text{vis}}$ **do**

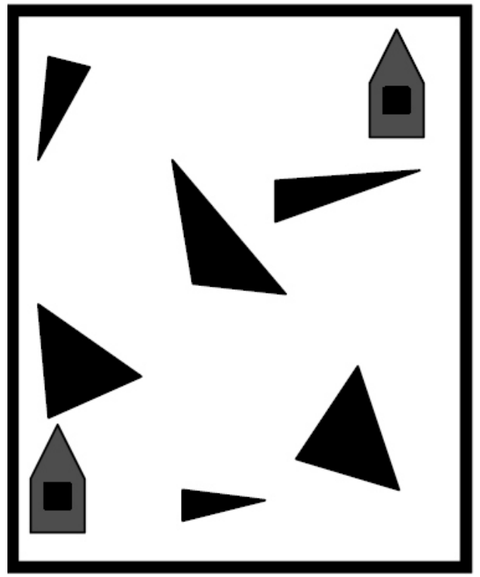
3: $w(uv) \leftarrow |uv|$

4: **return** $\textsc{Dijkstra}(G_{\text{vis}}, w, s, t)$ $O(n^2 \log n)$

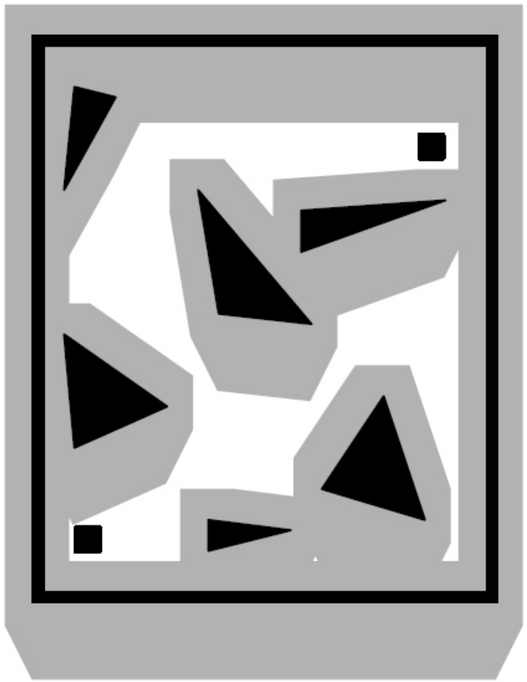
Theorem 2: A shortest collision-free st-path in a region with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Shortest path for polygonal robots

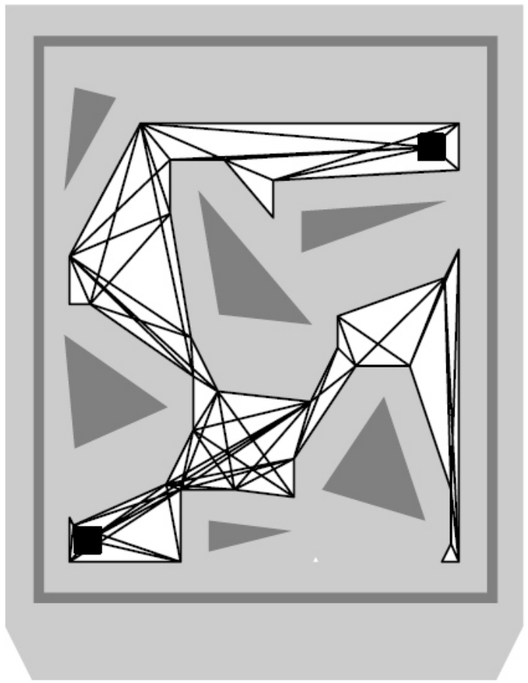
work space



configuration space

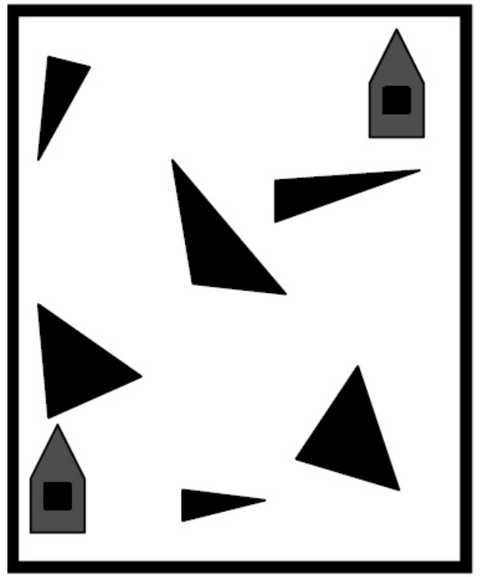


visibility graph

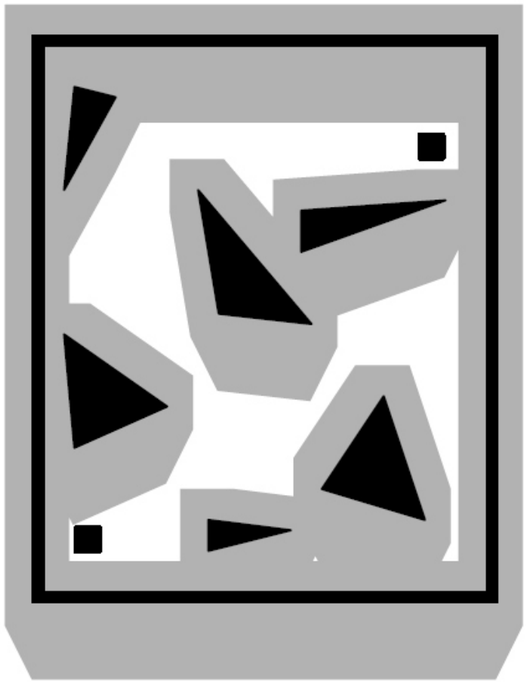


Shortest path for polygonal robots

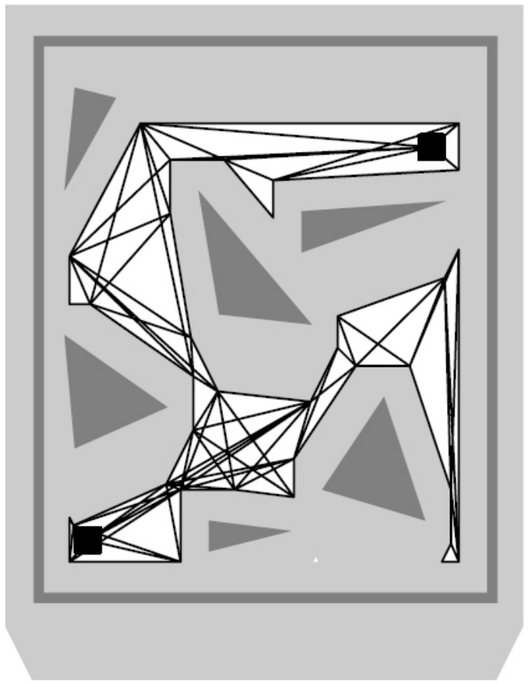
work space



configuration space



visibility graph

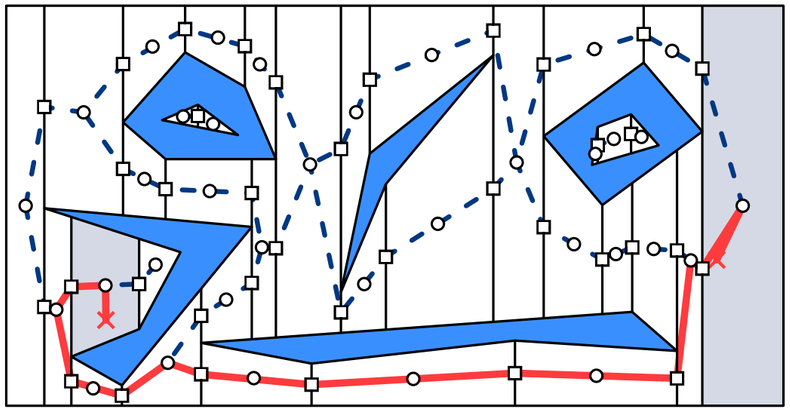


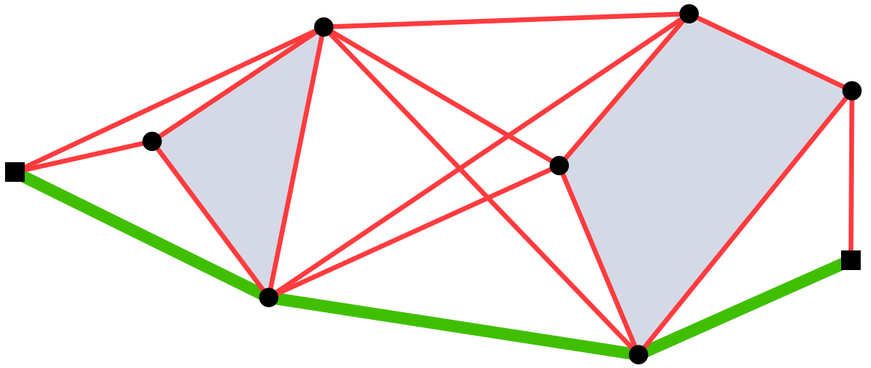
Theorem: A shortest st-path for a convex, polygonal, translating robot of constant complexity in a region with polygonal obstacles with n edges can be computed in $O(n^2 \log n)$ time.

Summary

We have seen several approaches, all using graphs in the free space:

- Vertical decomposition for arbitrary paths
- Visibility graph for shortest paths

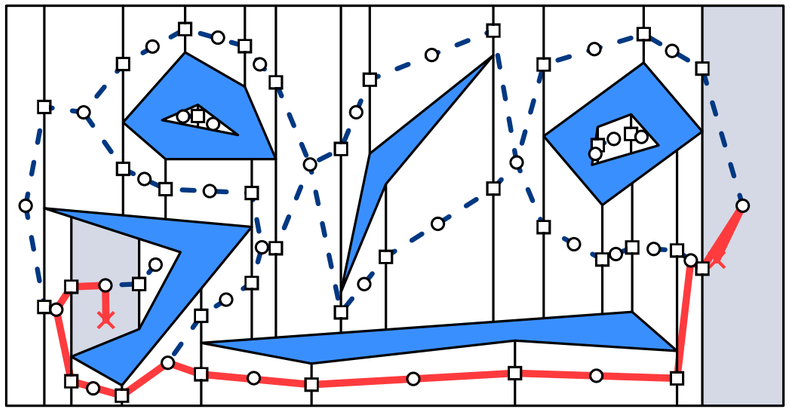


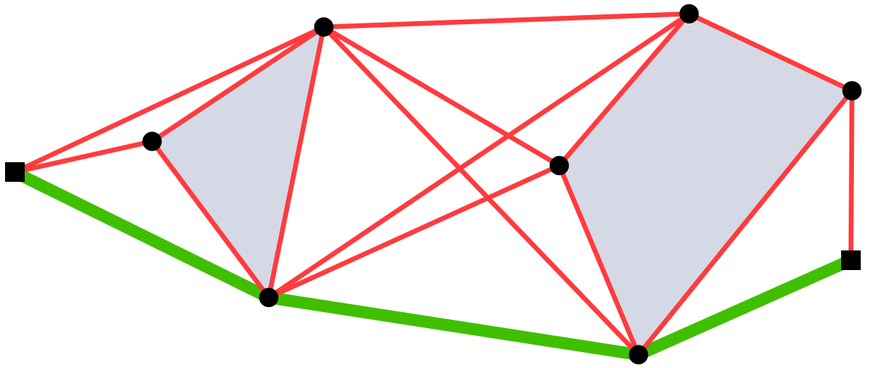


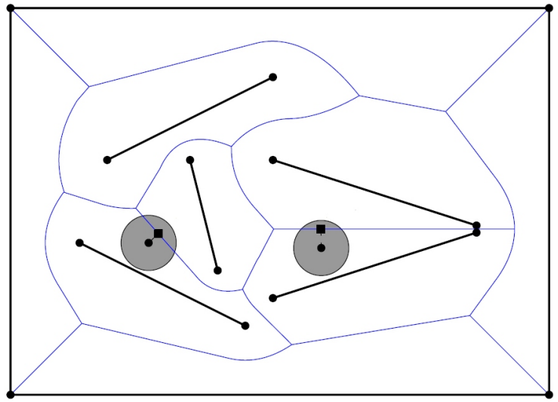
Summary

We have seen several approaches, all using graphs in the free space:

- Vertical decomposition for arbitrary paths
- Visibility graph for shortest paths
- Voronoi diagram for middle paths



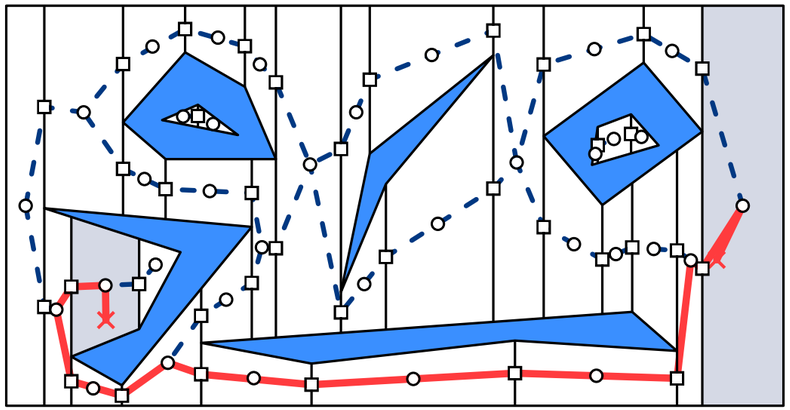


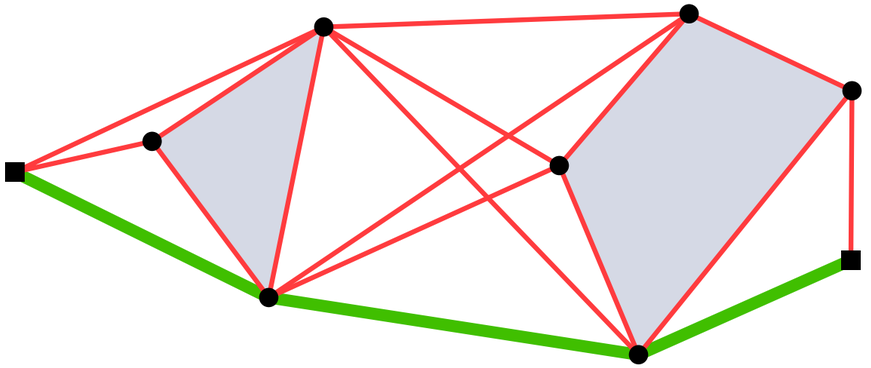


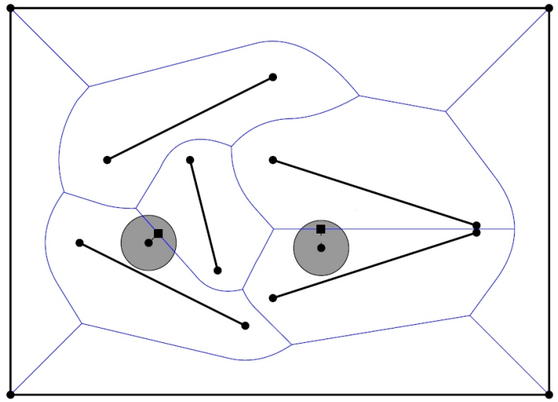
Summary

We have seen several approaches, all using graphs in the free space:

- Vertical decomposition for arbitrary paths
- Visibility graph for shortest paths
- Voronoi diagram for middle paths







https://youtu.be/u0BfGtRuNZA