Computational Geometry

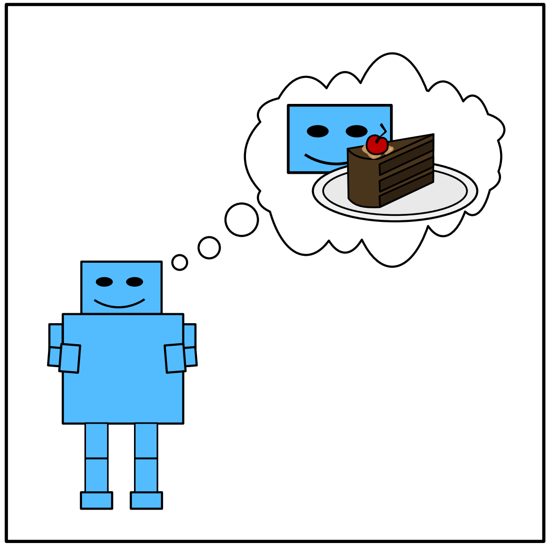
Lecture 10:
Motion Planning

Part I:
Point-Shaped Robots

Philipp Kindermann

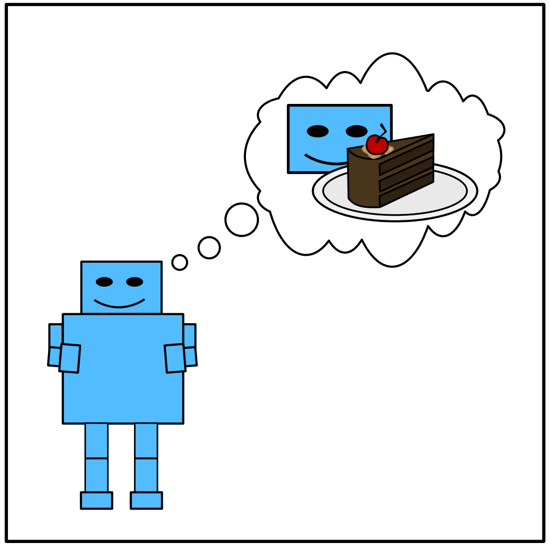
Winter Semester 2020

Planning



current situation,
desired situation

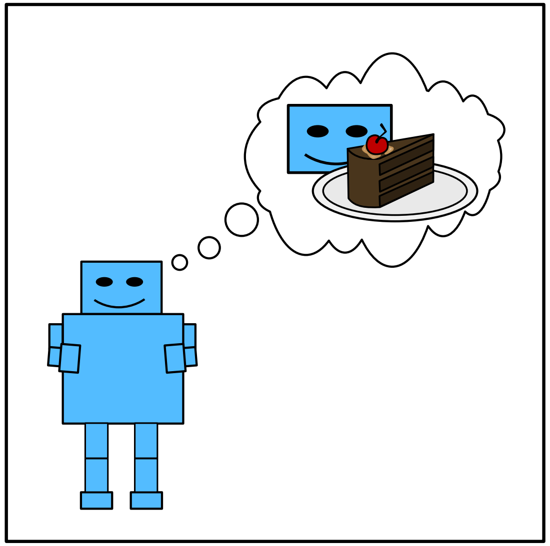
Planning



current situation,
desired situation

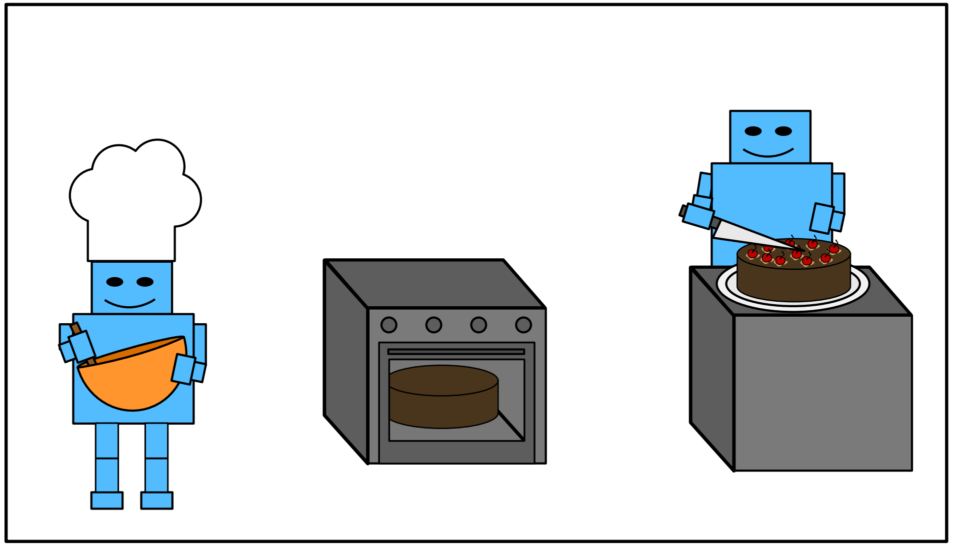
$\Rightarrow$

Planning



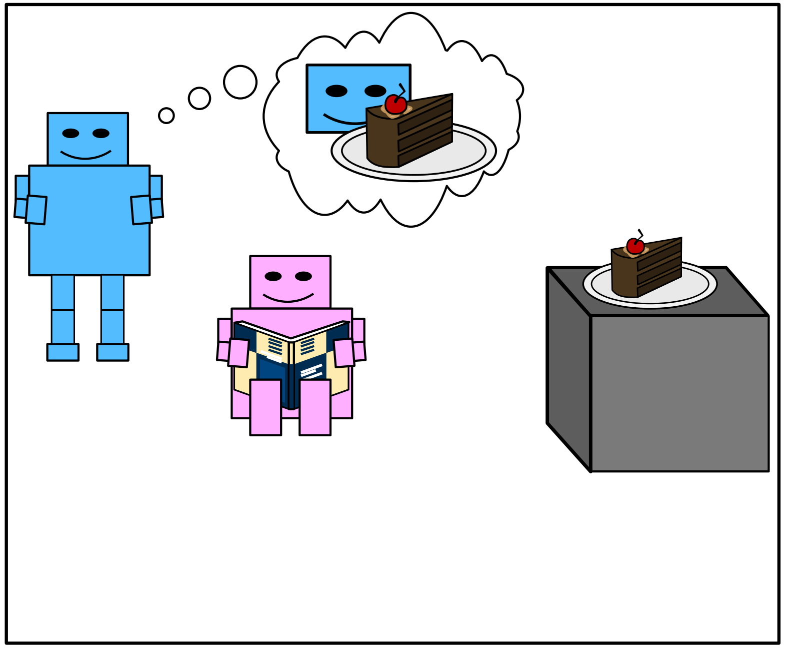
current situation,
desired situation

$\Rightarrow$



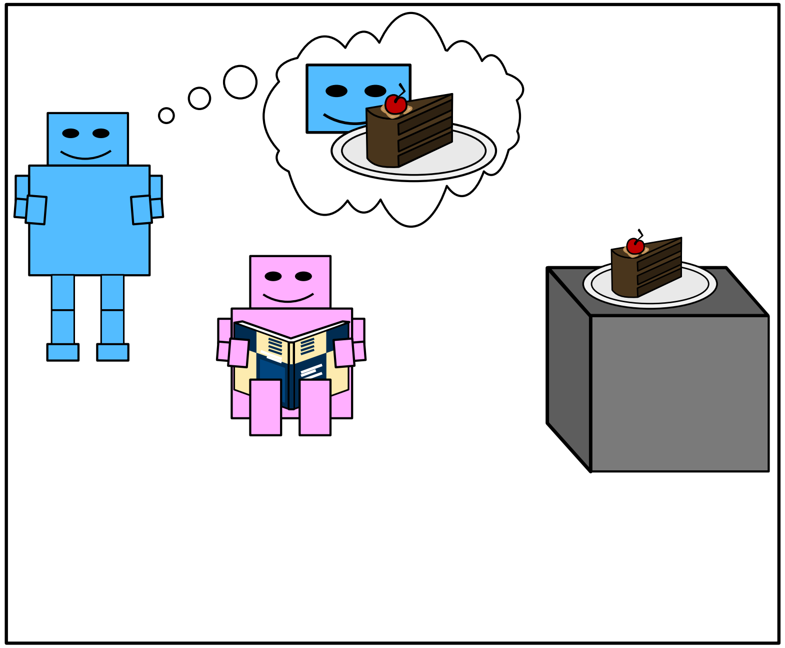
sequence of steps to reach
the one from the other

Path Planning



current location,
desired location

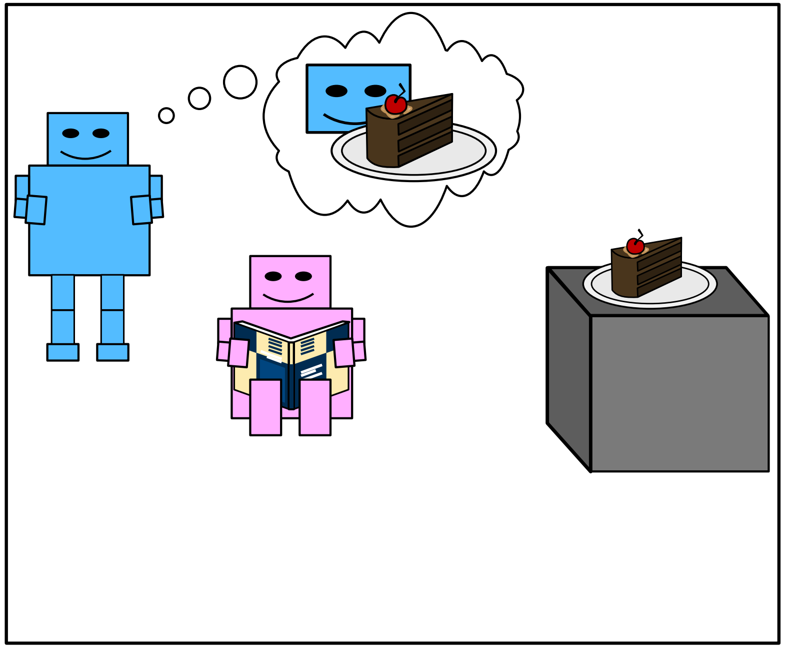
Path Planning



current location,
desired location

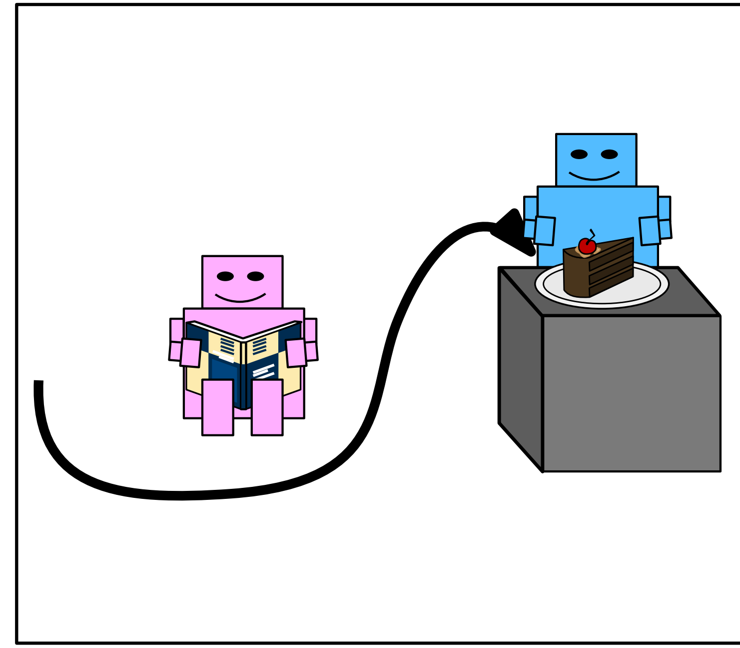
$\Rightarrow$

Path Planning



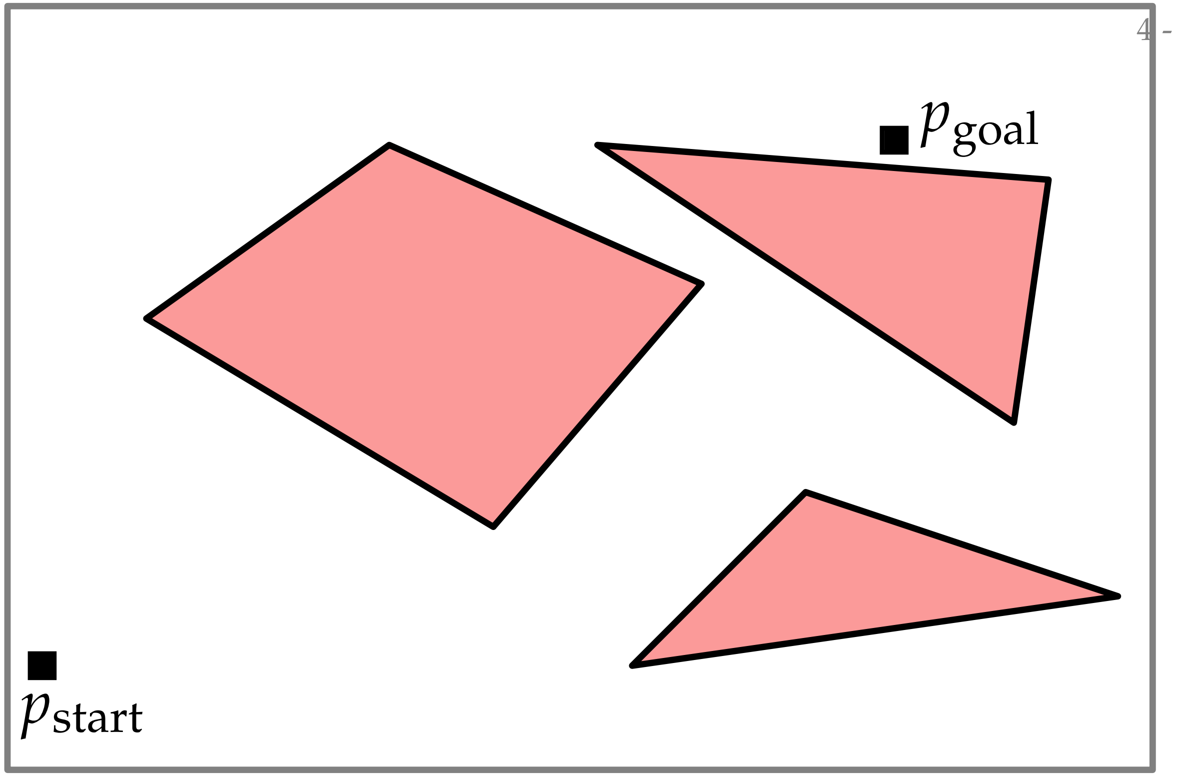
current location,
desired location

$\Rightarrow$

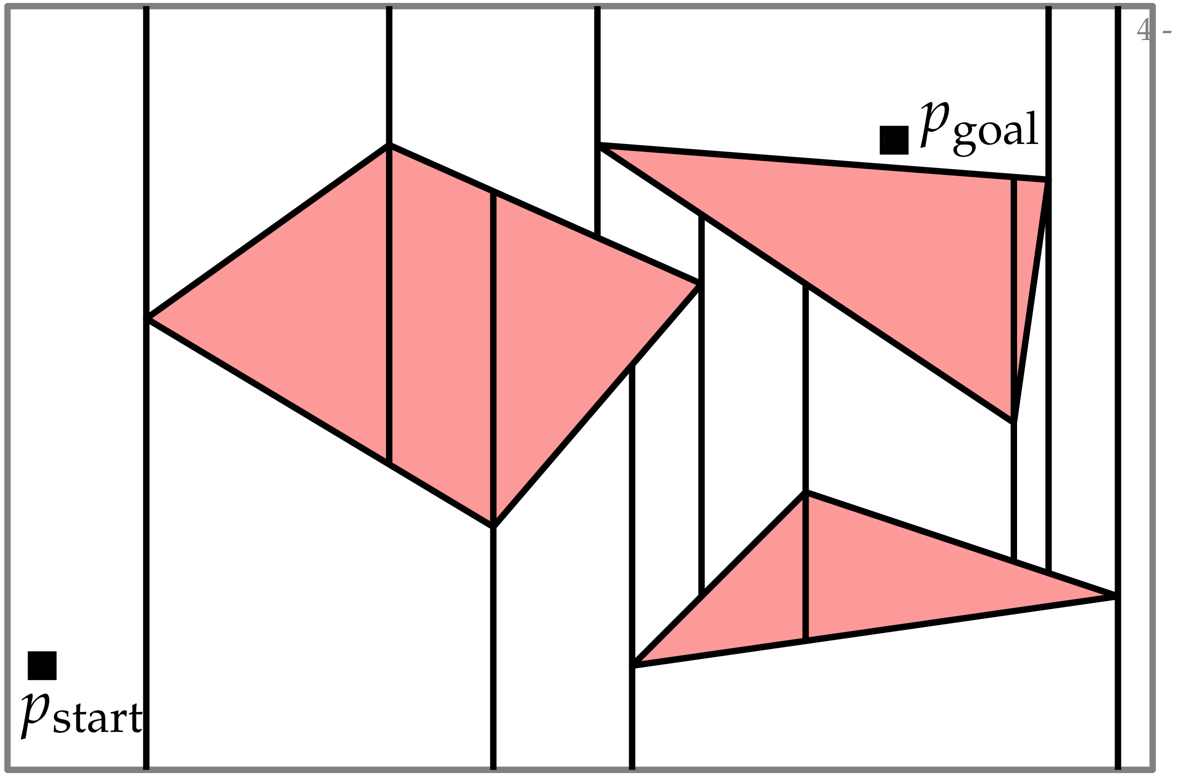


path to reach the
one from the other

Point-Shaped Robots

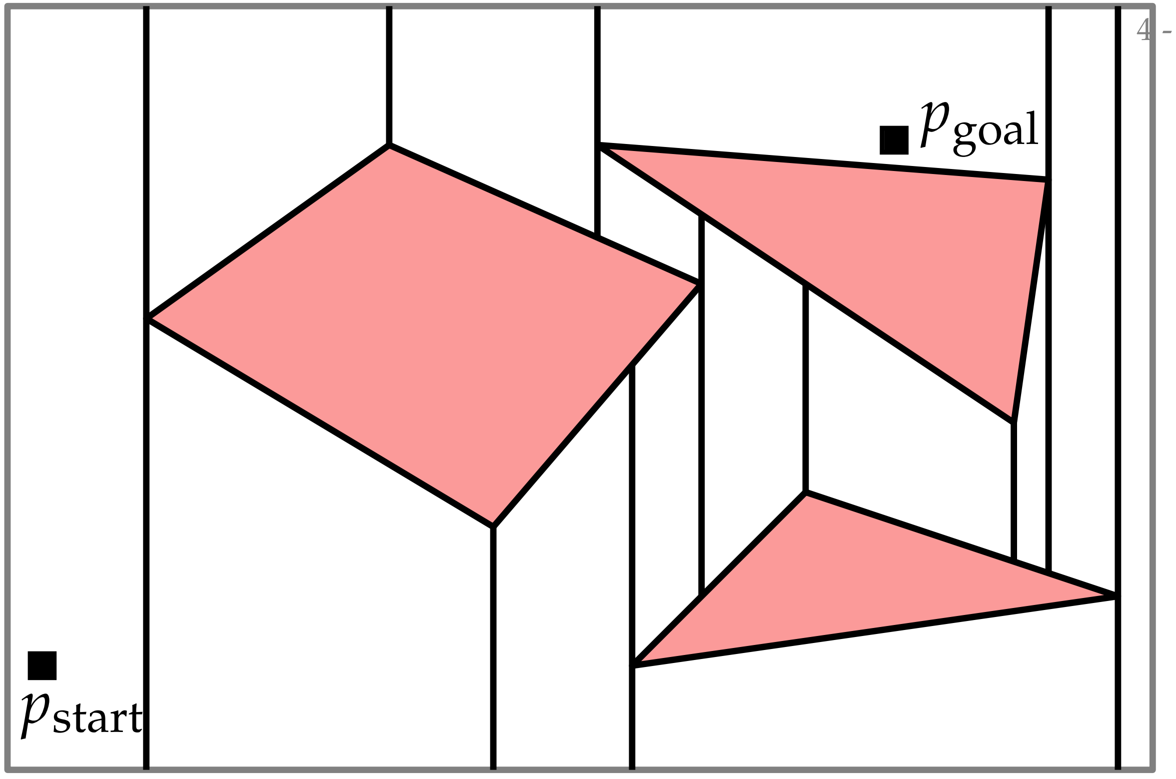


Point-Shaped Robots



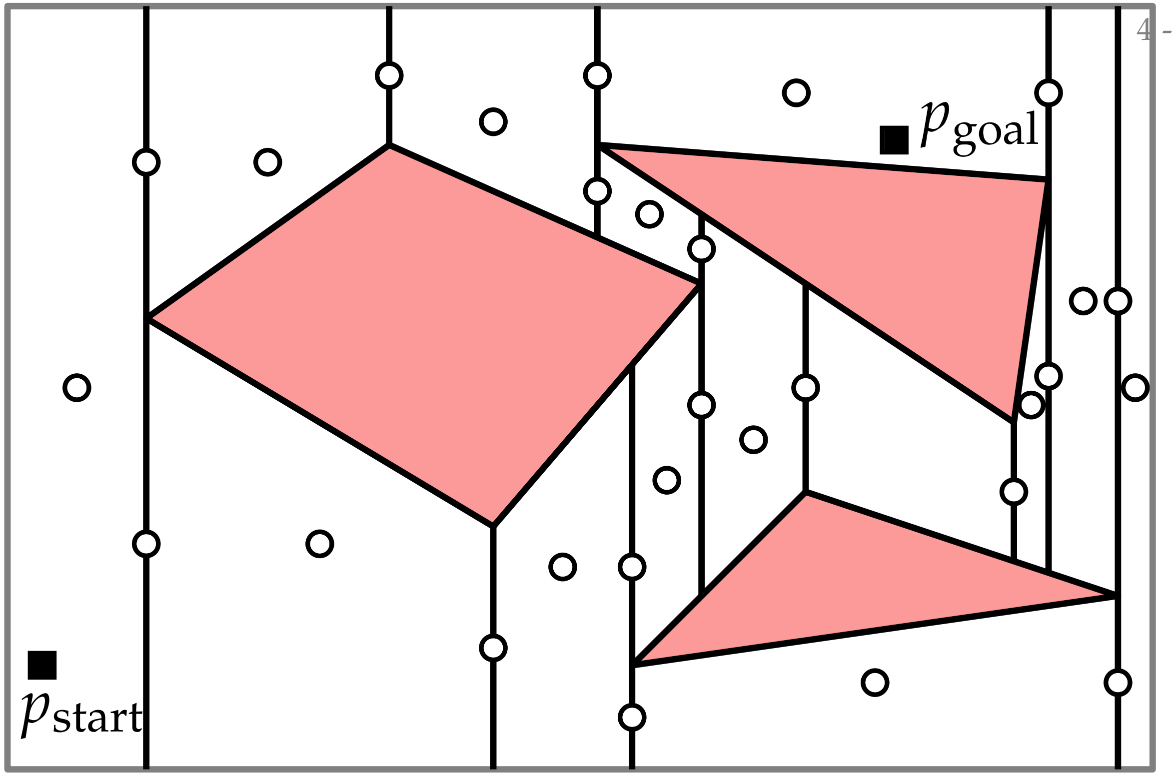
- Create trapezoidal map of obstacle edges.

Point-Shaped Robots



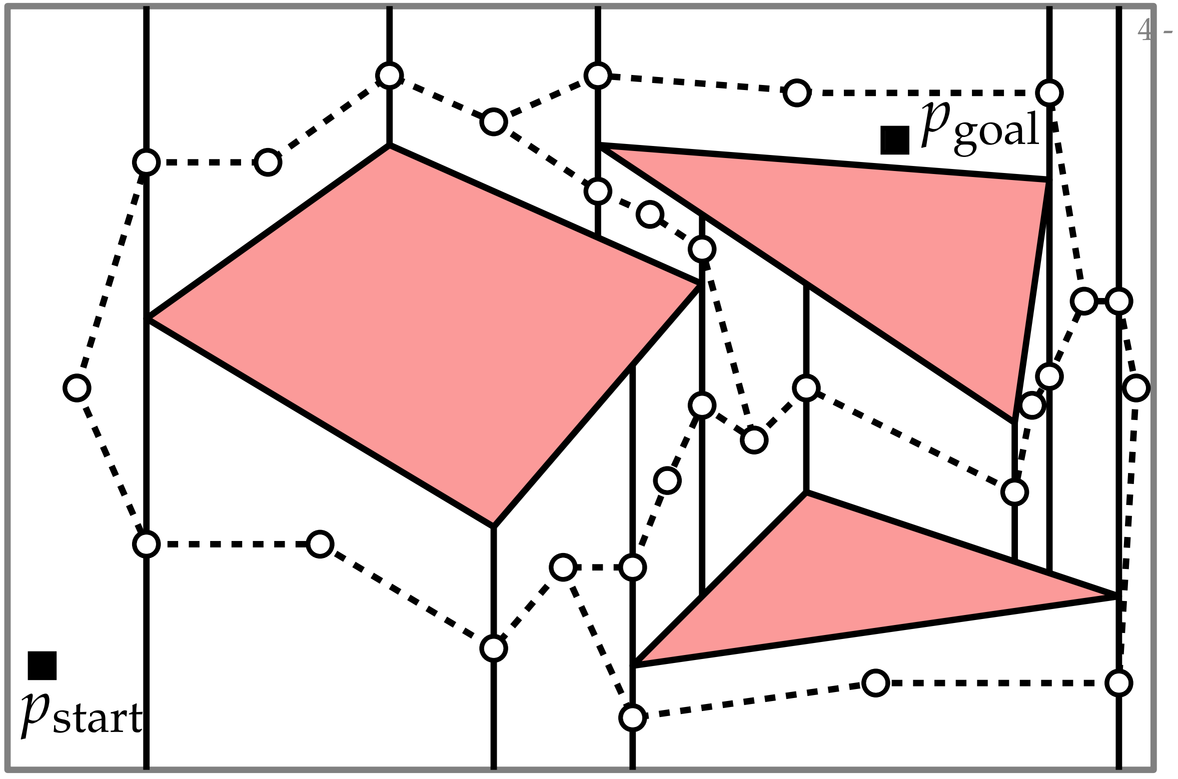
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.

Point-Shaped Robots



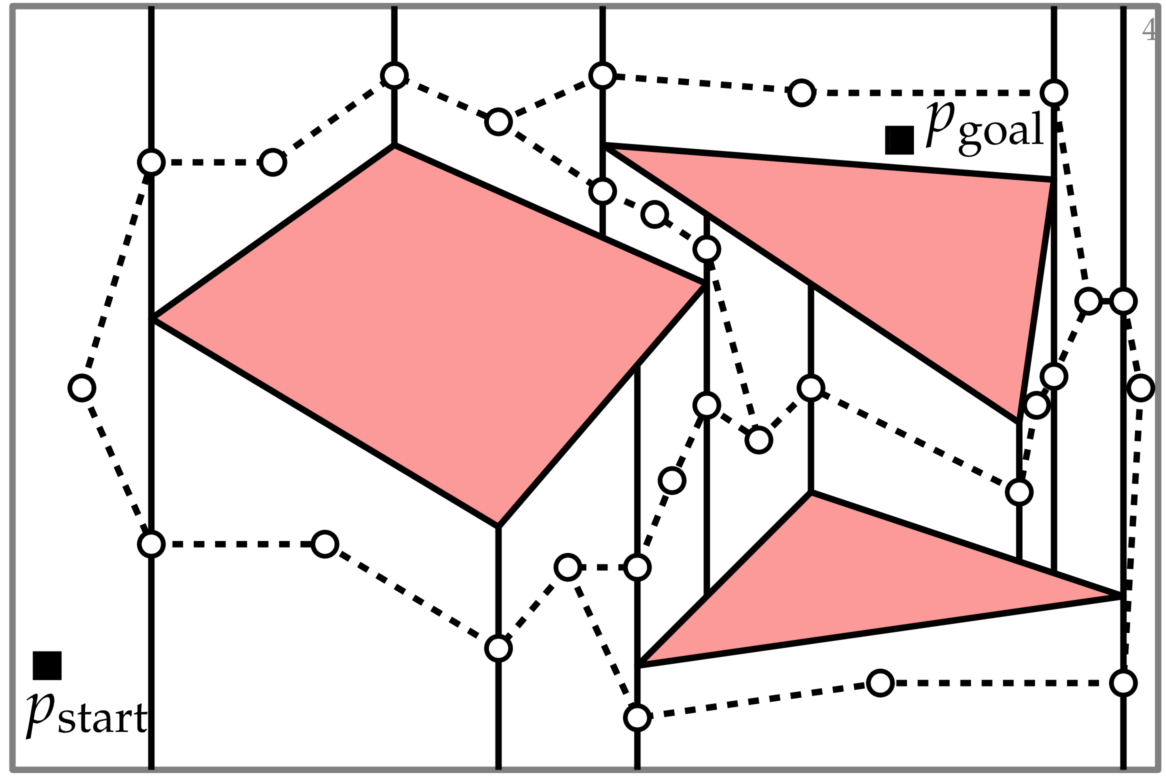
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.

Point-Shaped Robots



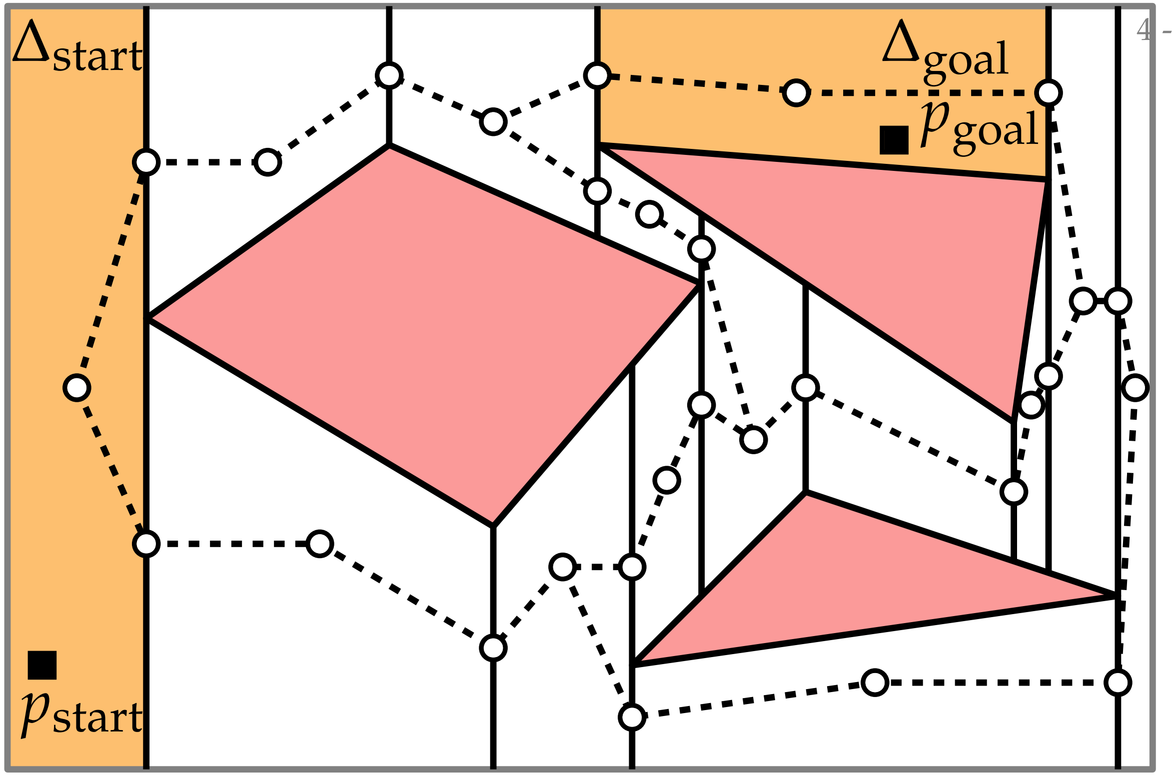
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect “neighboring” vertices by line segm.

Point-Shaped Robots



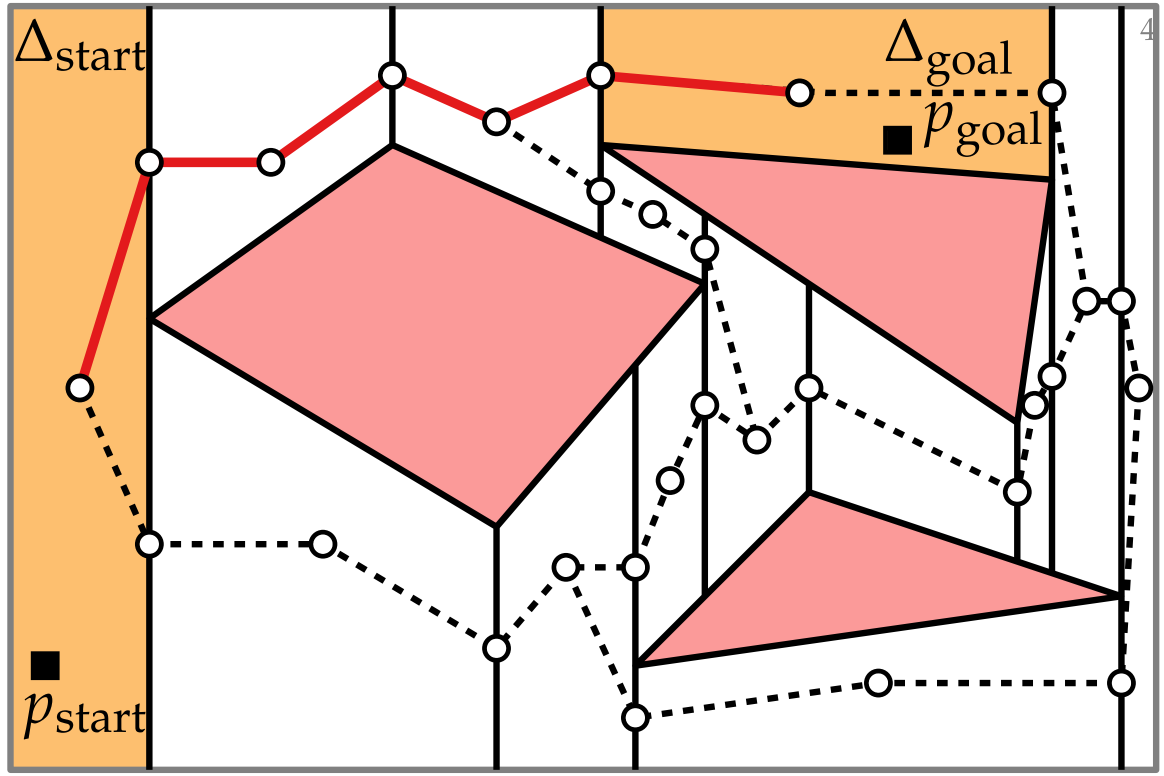
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect “neighboring” vertices by line segm.
- Locate p_{start}, p_{goal} in map

Point-Shaped Robots



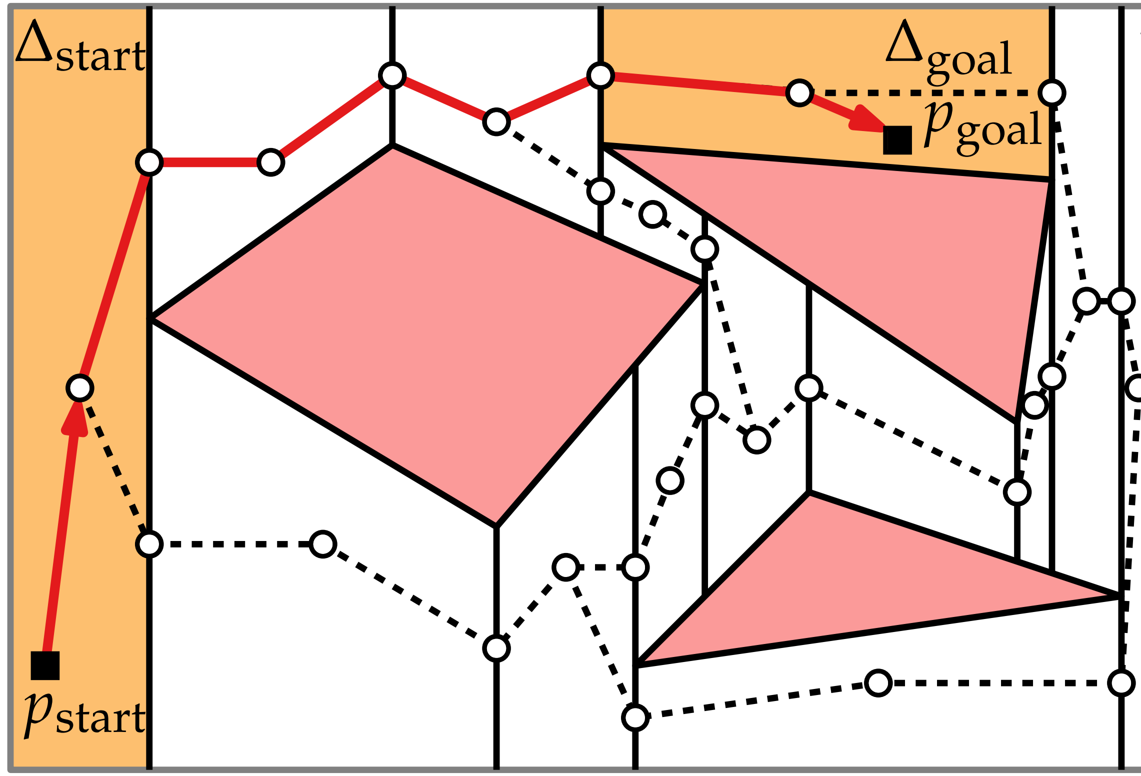
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect "neighboring" vertices by line segm.
- Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\rightarrow \Delta_{\text{start}}, \Delta_{\text{goal}}$.

Point-Shaped Robots



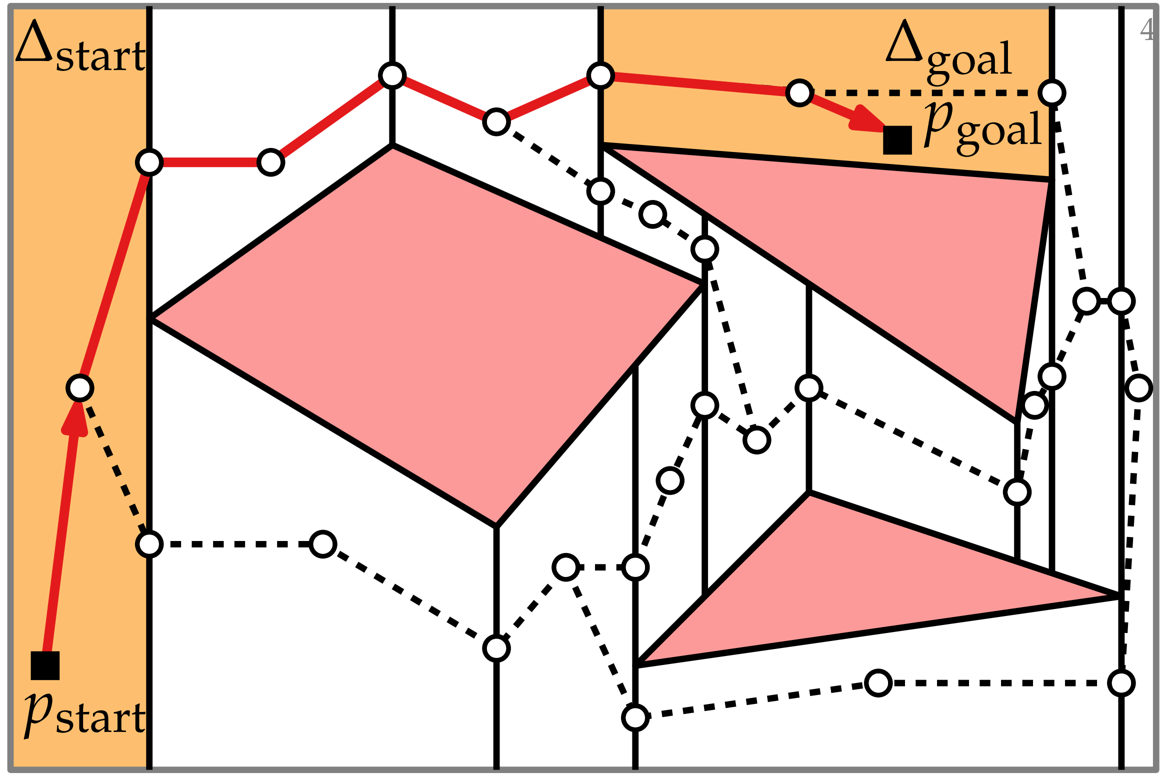
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect “neighboring” vertices by line segm.
- Locate p_{start}, p_{goal} in map $\to \Delta_{start}, \Delta_{goal}$.
- Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.

Point-Shaped Robots



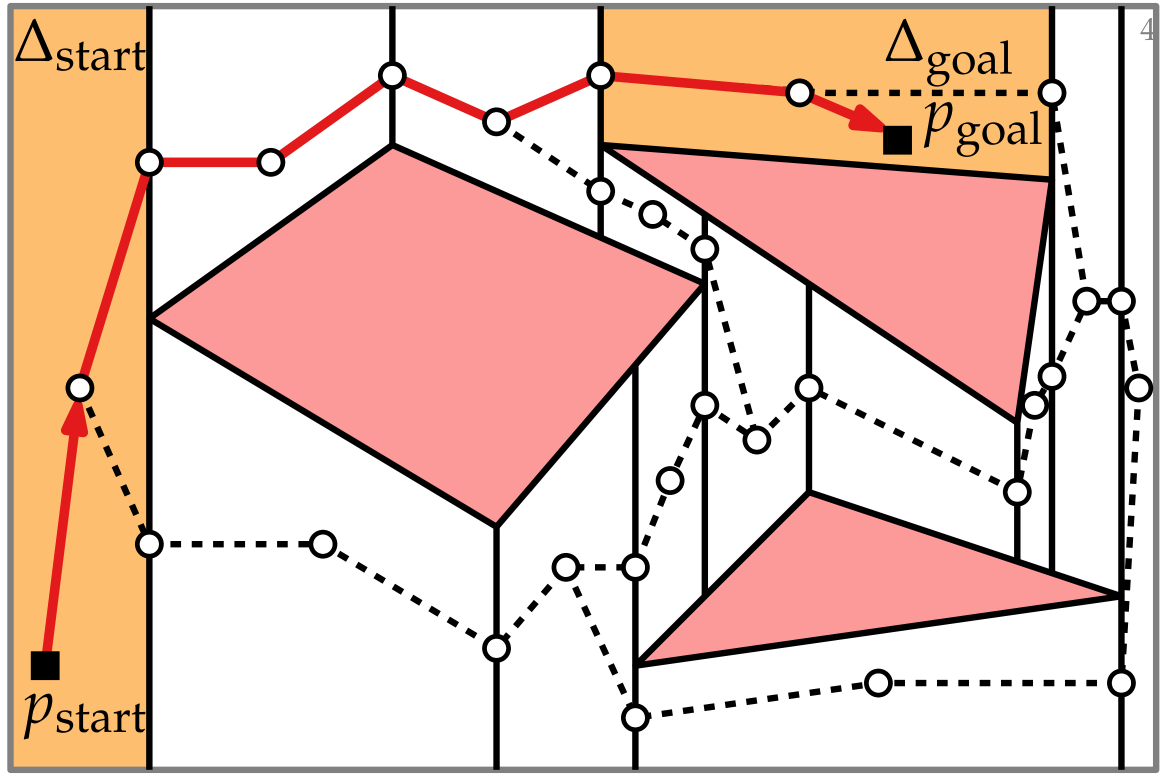
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect "neighboring" vertices by line segm.
- Locate p_{start}, p_{goal} in map $\to \Delta_{start}, \Delta_{goal}$.
- Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.
- Connect p_{start}, p_{goal} to π by line segments.

Point-Shaped Robots



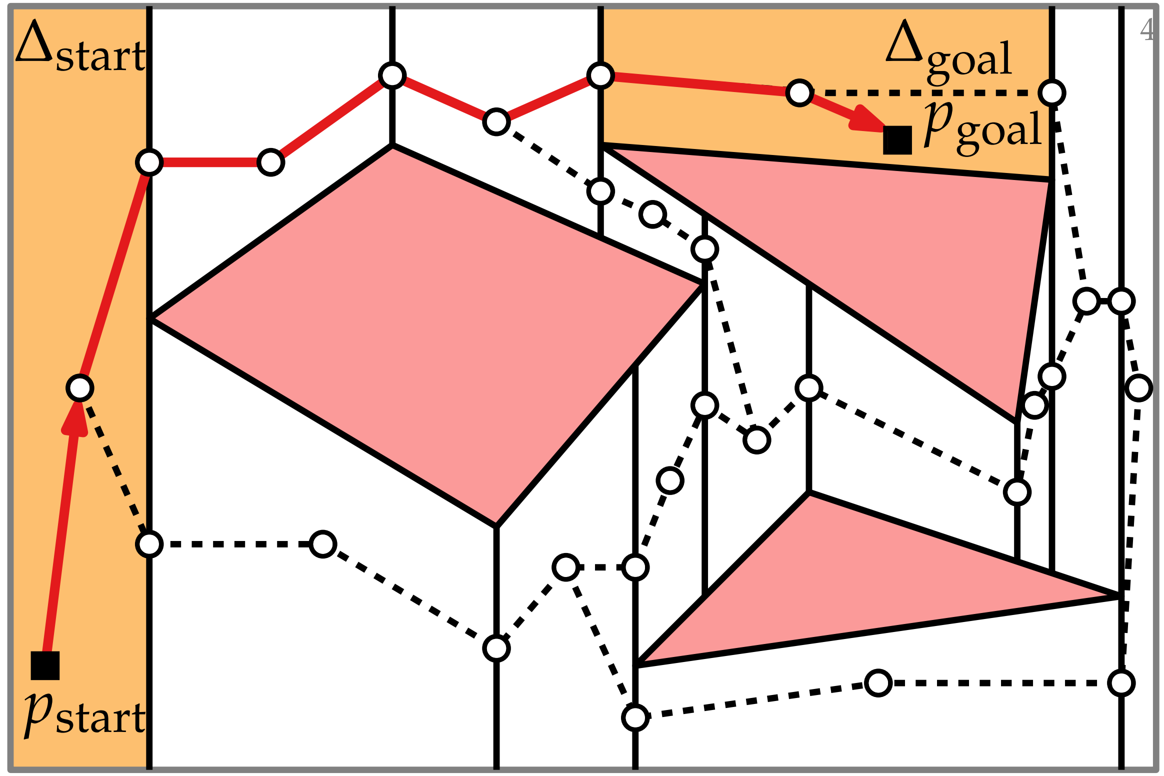
- Create trapezoidal map of obstacle edges.
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect "neighboring" vertices by line segm.
- Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\rightarrow \Delta_{\text{start}}, \Delta_{\text{goal}}$.
- Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.
- Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.

Point-Shaped Robots



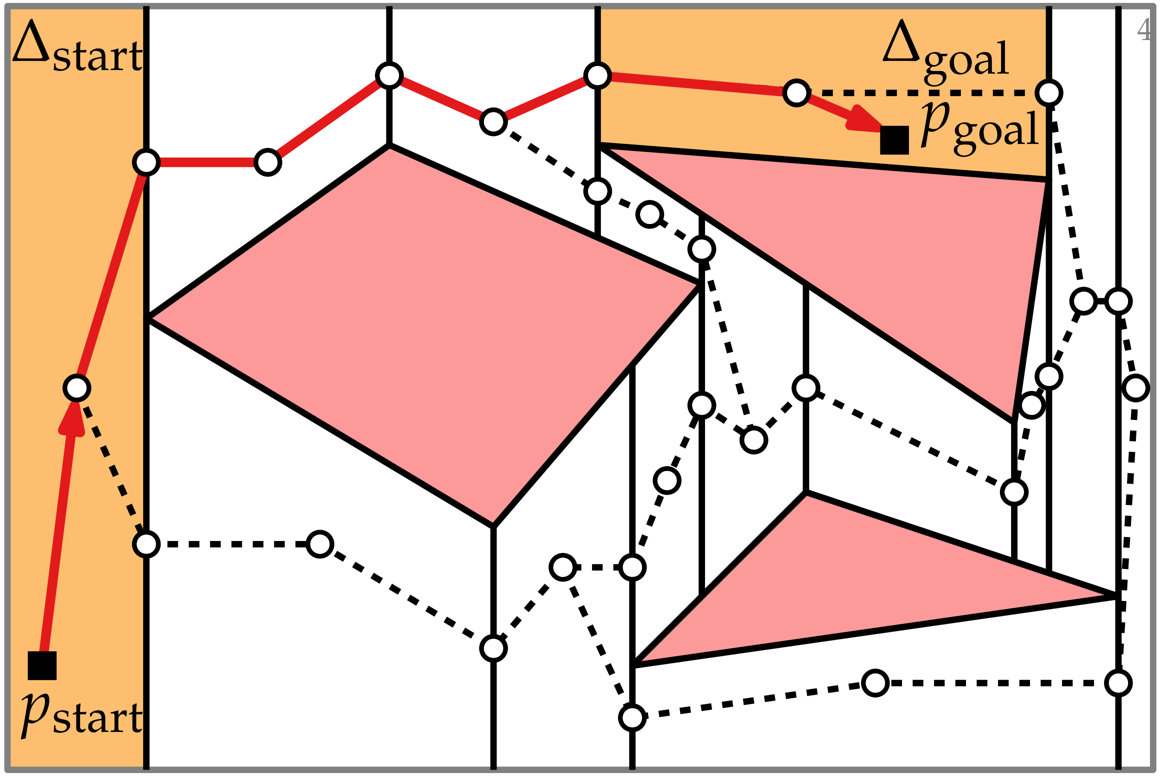
- Create trapezoidal map of obstacle edges. $O(n \log n)$
- Remove vertical extensions inside obstacles.
- Vertices at centers of trapez. and vertical ext.
- Connect "neighboring" vertices by line segm.
- Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\rightarrow \Delta_{\text{start}}, \Delta_{\text{goal}}$.
- Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.
- Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.

Point-Shaped Robots



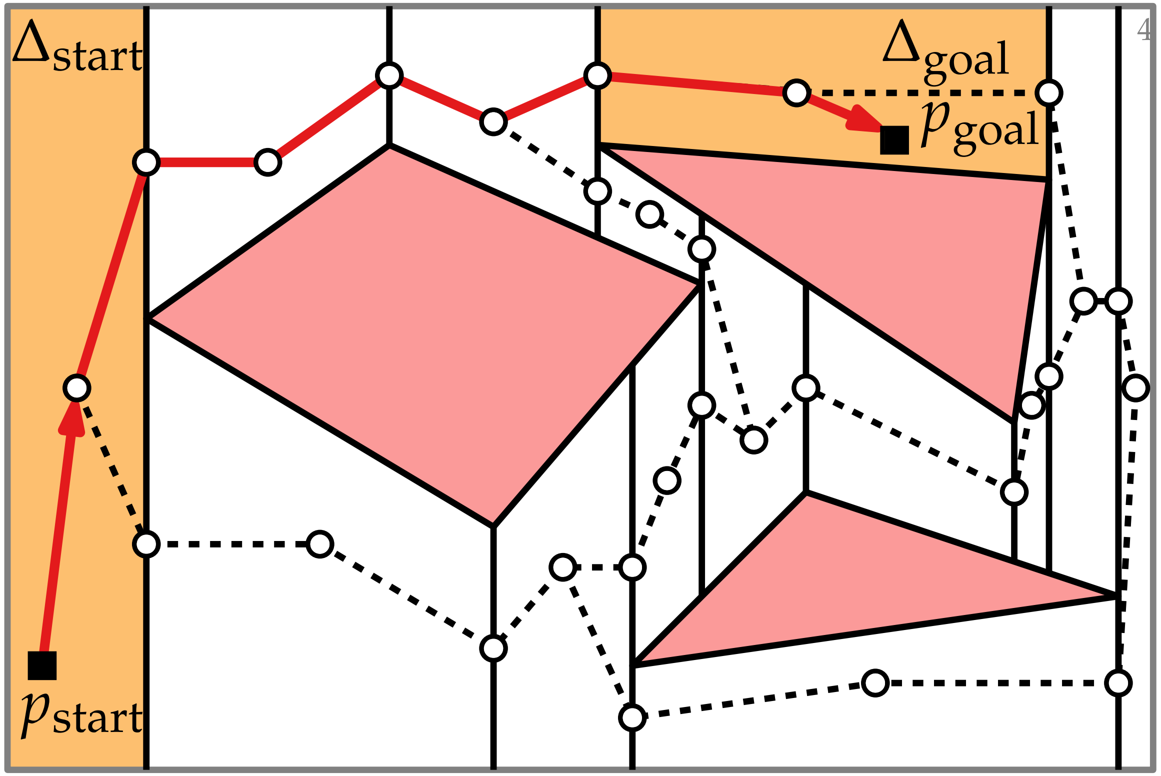
- Create trapezoidal map of obstacle edges. $O(n \log n)$
- Remove vertical extensions inside obstacles. $O(n)$
- Vertices at centers of trapez. and vertical ext.
- Connect "neighboring" vertices by line segm.
- Locate p_{start}, p_{goal} in map $\rightarrow \Delta_{start}, \Delta_{goal}$.
- Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.
- Connect p_{start}, p_{goal} to π by line segments.

Point-Shaped Robots



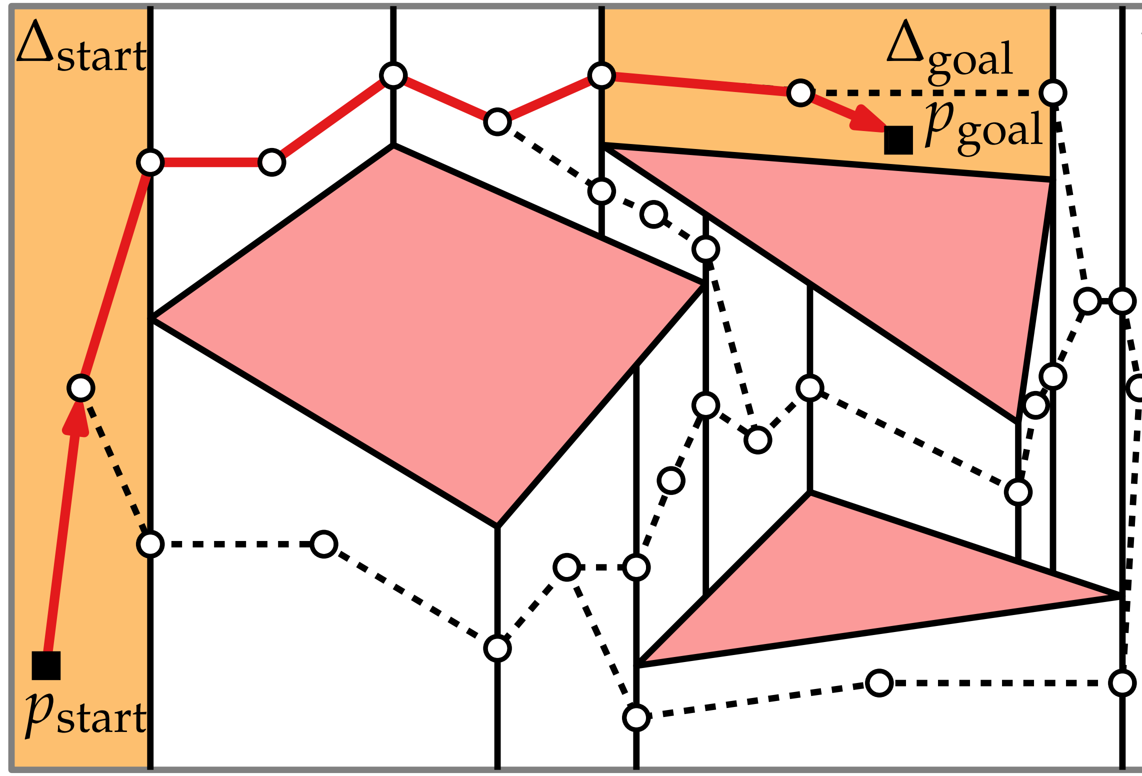
■ Create trapezoidal map of obstacle edges.	$O(n \log n)$
■ Remove vertical extensions inside obstacles.	$O(n)$
■ Vertices at centers of trapez. and vertical ext.	$O(n)$
■ Connect "neighboring" vertices by line segm.	
■ Locate p_{start}, p_{goal} in map $\rightarrow \Delta_{start}, \Delta_{goal}$.	
■ Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	
■ Connect p_{start}, p_{goal} to π by line segments.	

Point-Shaped Robots



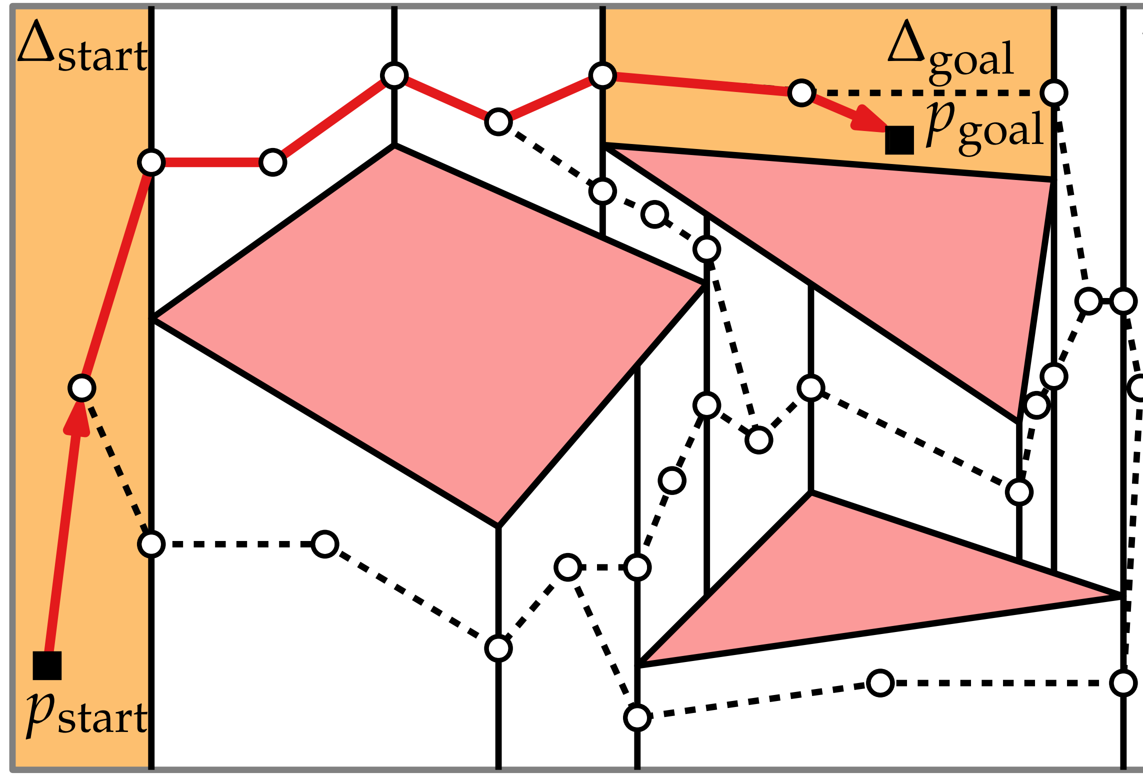
■ Create trapezoidal map of obstacle edges.	$O(n \log n)$
■ Remove vertical extensions inside obstacles.	$O(n)$
■ Vertices at centers of trapez. and vertical ext.	$O(n)$
■ Connect "neighboring" vertices by line segm.	$O(n)$
■ Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\rightarrow \Delta_{\text{start}}, \Delta_{\text{goal}}$.	
■ Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	
■ Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.	

Point-Shaped Robots



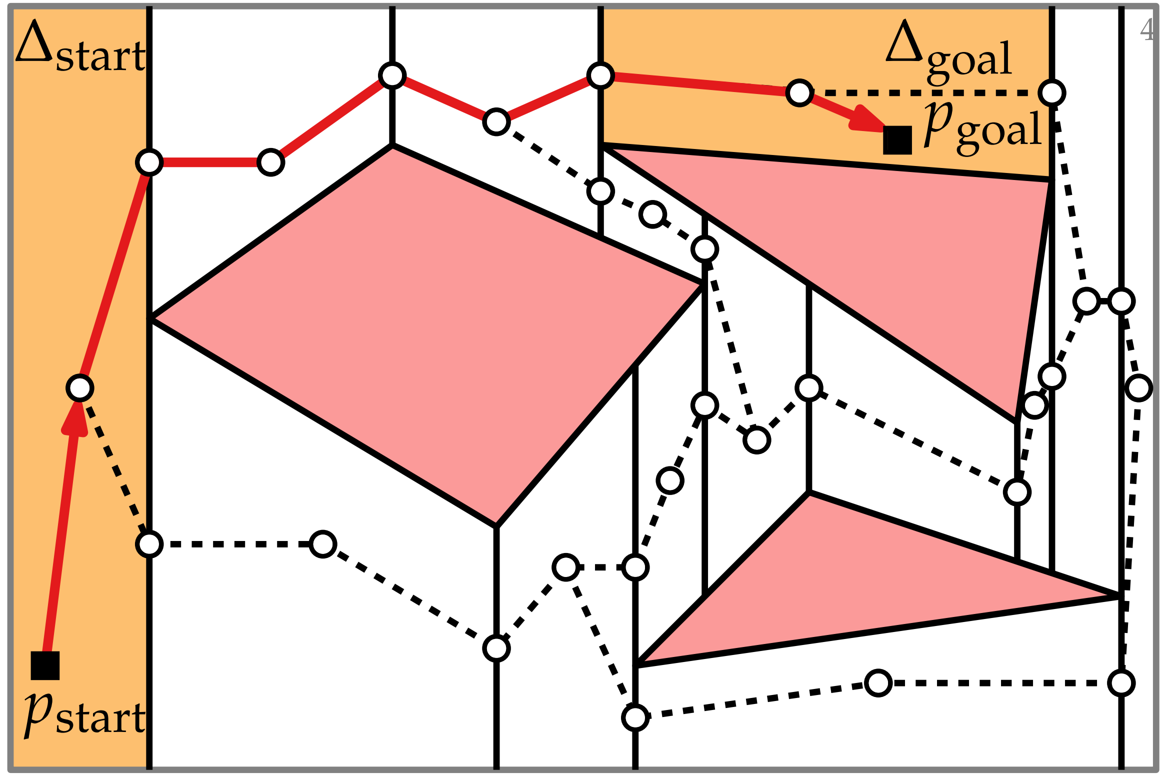
Create trapezoidal map of obstacle edges.	$O(n \log n)$
Remove vertical extensions inside obstacles.	$O(n)$
Vertices at centers of trapez. and vertical ext.	$O(n)$
Connect "neighboring" vertices by line segm.	$O(n)$
Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\rightarrow \Delta_{\text{start}}, \Delta_{\text{goal}}$.	$O(\log n)$
Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	
Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.	

Point-Shaped Robots



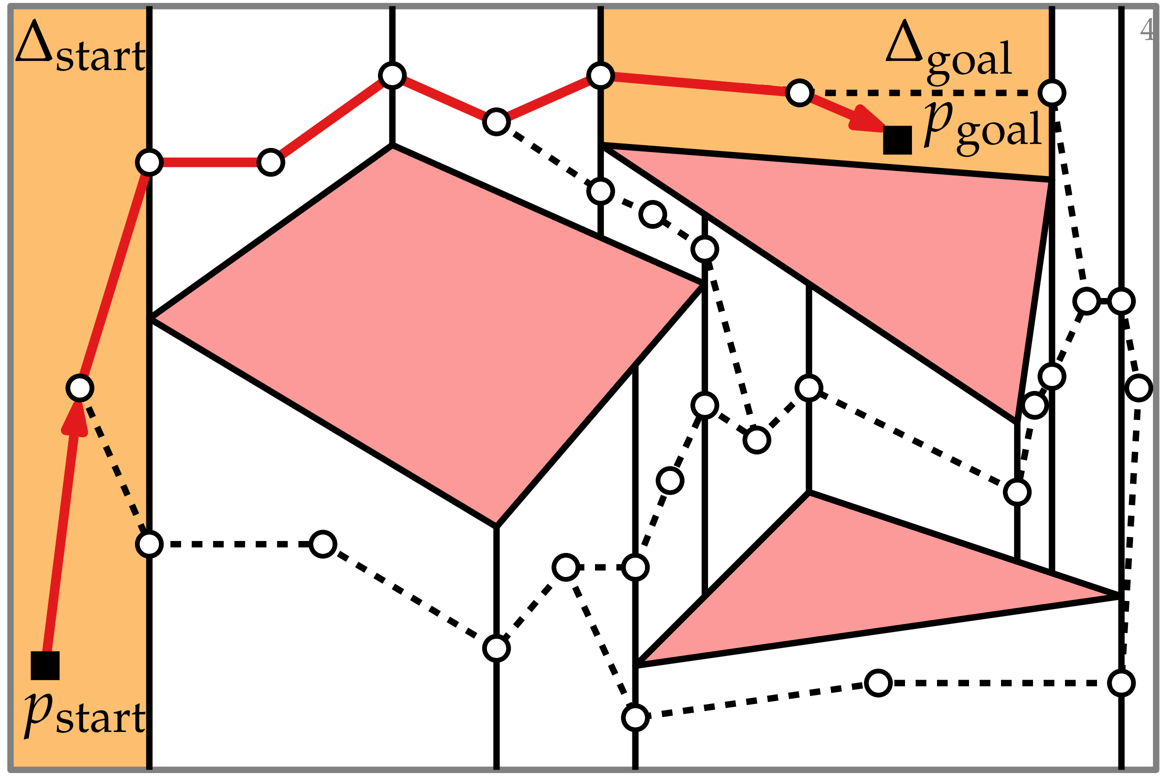
Create trapezoidal map of obstacle edges.	$O(n \log n)$
Remove vertical extensions inside obstacles.	$O(n)$
Vertices at centers of trapez. and vertical ext.	$O(n)$
Connect "neighboring" vertices by line segm.	$O(n)$
Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\to \Delta_{\text{start}}, \Delta_{\text{goal}}$.	$O(\log n)$
Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	$O(n)$
Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.	

Point-Shaped Robots



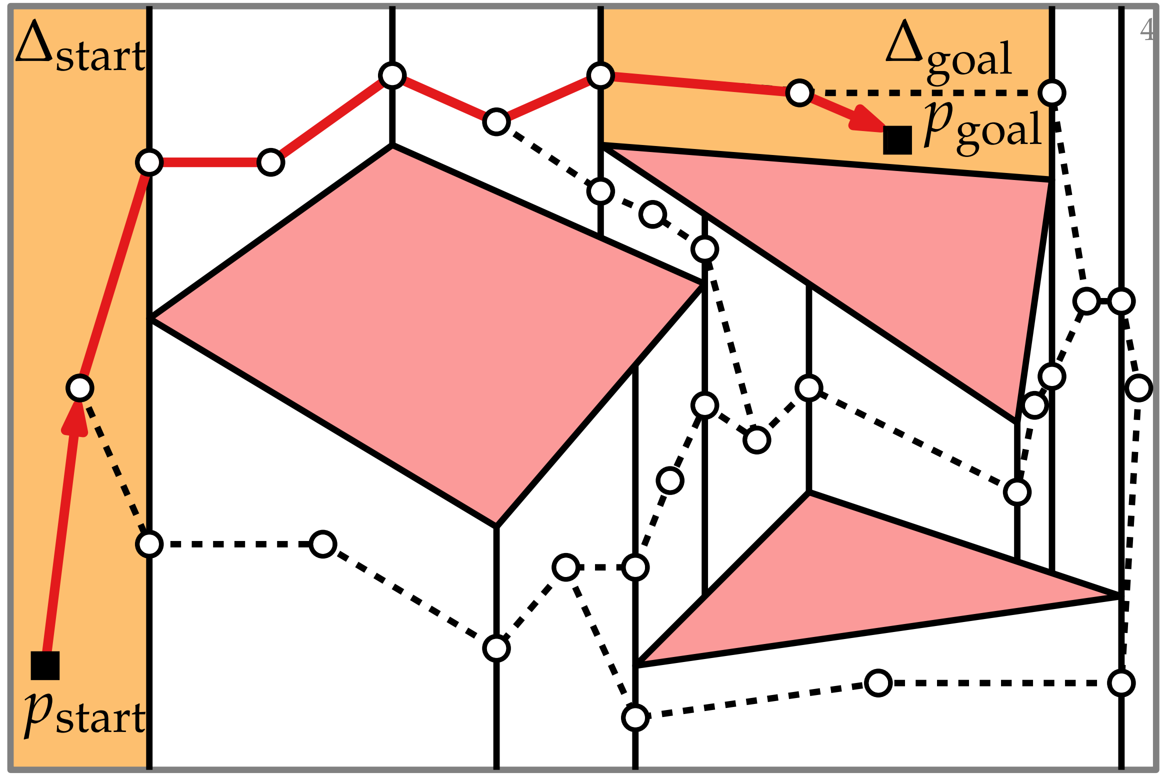
■ Create trapezoidal map of obstacle edges.	$O(n \log n)$
■ Remove vertical extensions inside obstacles.	$O(n)$
■ Vertices at centers of trapez. and vertical ext.	$O(n)$
■ Connect "neighboring" vertices by line segm.	$O(n)$
■ Locate p_{start}, p_{goal} in map $\to \Delta_{start}, \Delta_{goal}$.	$O(\log n)$
■ Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	$O(n)$
■ Connect p_{start}, p_{goal} to π by line segments.	$O(1)$

Point-Shaped Robots



Create trapezoidal map of obstacle edges.	$O(n \log n)$
Remove vertical extensions inside obstacles.	$O(n)$
Vertices at centers of trapez. and vertical ext.	$O(n)$
Connect "neighboring" vertices by line segm.	$O(n)$
Locate $p_{\text{start}}, p_{\text{goal}}$ in map $\to \Delta_{\text{start}}, \Delta_{\text{goal}}$.	$O(\log n)$
Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	$O(n)$
Connect $p_{\text{start}}, p_{\text{goal}}$ to π by line segments.	$O(1)$

Point-Shaped Robots



preprocessing	Create trapezoidal map of obstacle edges.	$O(n \log n)$
	Remove vertical extensions inside obstacles.	$O(n)$
	Vertices at centers of trapez. and vertical ext.	$O(n)$
	Connect "neighboring" vertices by line segm.	$O(n)$
querying	Locate p_{start}, p_{goal} in map $\rightarrow \Delta_{start}, \Delta_{goal}$.	$O(\log n)$
	Do breadth-first search in the *roadmap* to find a path π from Δ_{start} to Δ_{goal}.	$O(n)$
	Connect p_{start}, p_{goal} to π by line segments.	$O(1)$

A First Result

Theorem. We can preprocess a set of polygonal obstacles with a total of n edges in $O(n \log n)$ expected time such that, given a start and a goal position, we can find a collision-free path for a point robot in $O(n)$ time if it exists.

A First Result

Theorem. We can preprocess a set of polygonal obstacles with a total of n edges in $O(n \log n)$ expected time such that, given a start and a goal position, we can find a collision-free path for a point robot in $O(n)$ time if it exists.

A First Result

Theorem. We can preprocess a set of polygonal obstacles with a total of n edges in $O(n \log n)$ expected time such that, given a start and a goal position, we can find a collision-free path for a point robot in $O(n)$ time if it exists.

What about, say, *polygonal* robots?

Computational Geometry

Lecture 10:
Motion Planning

Part II:
Configuration Space

Philipp Kindermann

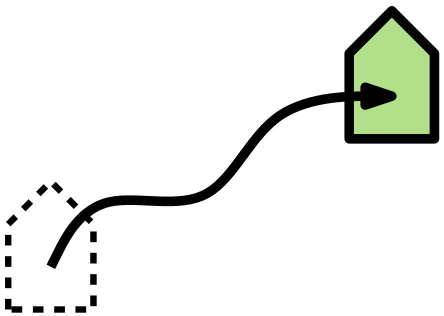
Winter Semester 2020

Degrees of Freedom

Every robot has some number *d* of *degrees of freedom*, meaning that its *configuration* with respect to the world can be specified by *d* parameters.

Degrees of Freedom

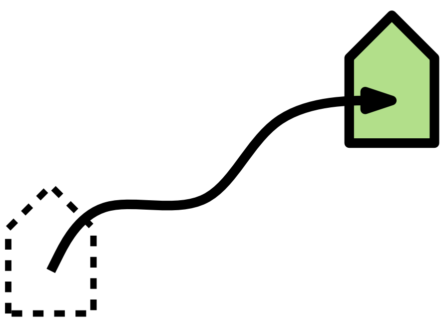
Every robot has some number *d* of *degrees of freedom*, meaning that its *configuration* with respect to the world can be specified by *d* parameters.



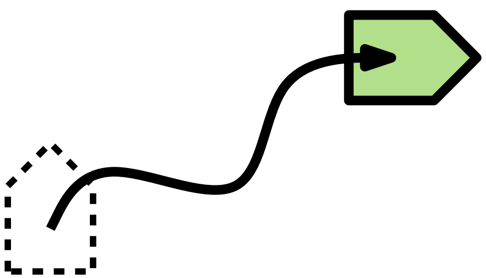
2D translating robot

Degrees of Freedom

Every robot has some number *d* of *degrees of freedom*, meaning that its *configuration* with respect to the world can be specified by *d* parameters.



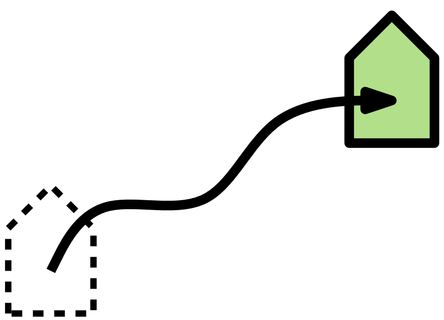
2D translating robot



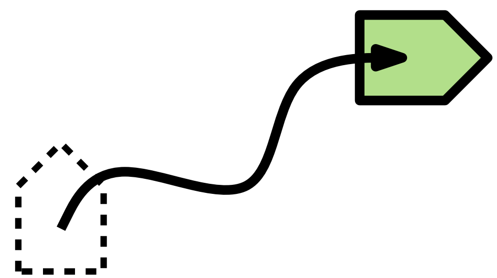
2D translating, rotating robot

Degrees of Freedom

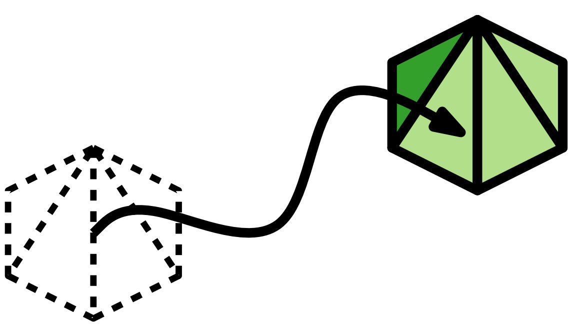
Every robot has some number *d* of *degrees of freedom*, meaning that its *configuration* with respect to the world can be specified by *d* parameters.



2D translating robot



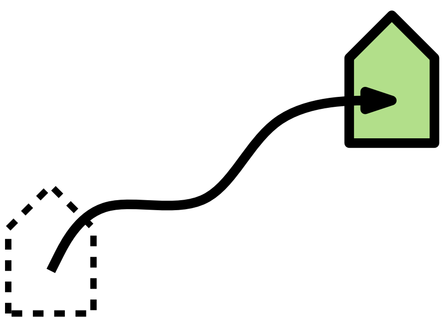
2D translating, rotating robot



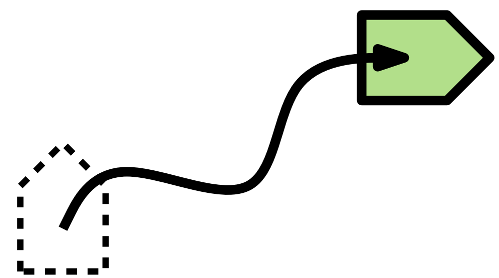
3D translating robot

Degrees of Freedom

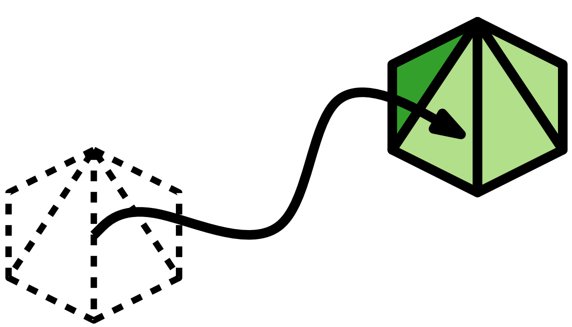
Every robot has some number *d* of *degrees of freedom*, meaning that its *configuration* with respect to the world can be specified by *d* parameters.



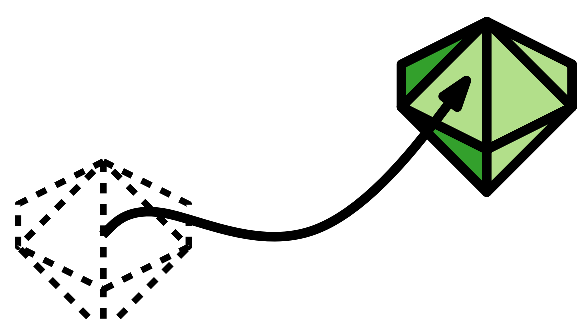
2D translating robot



2D translating, rotating robot

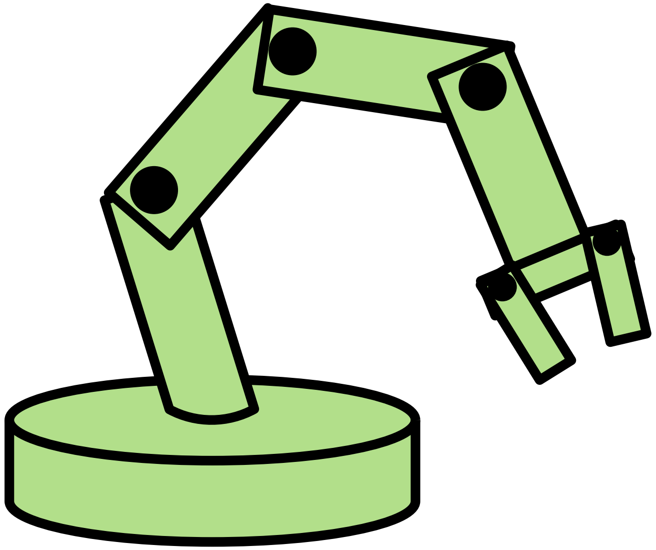


3D translating robot



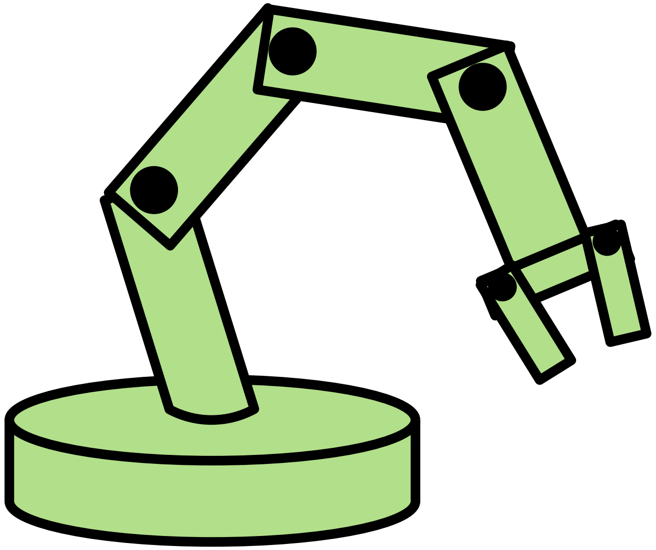
3D translating, rotating robot

Configuration Space



robotic arm

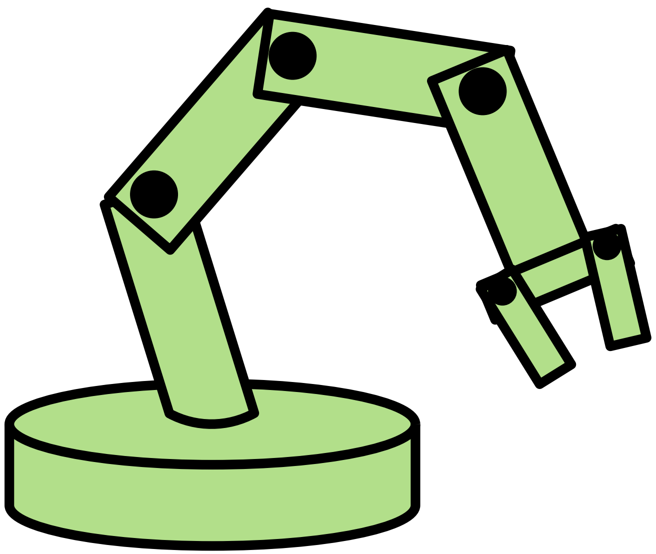
Configuration Space



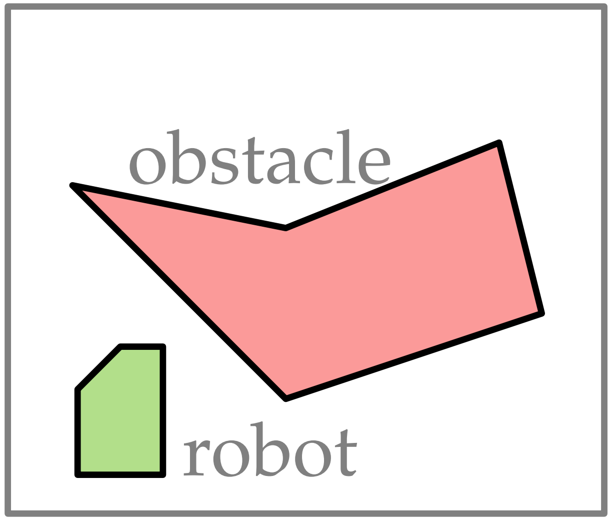
robotic arm

The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Configuration Space



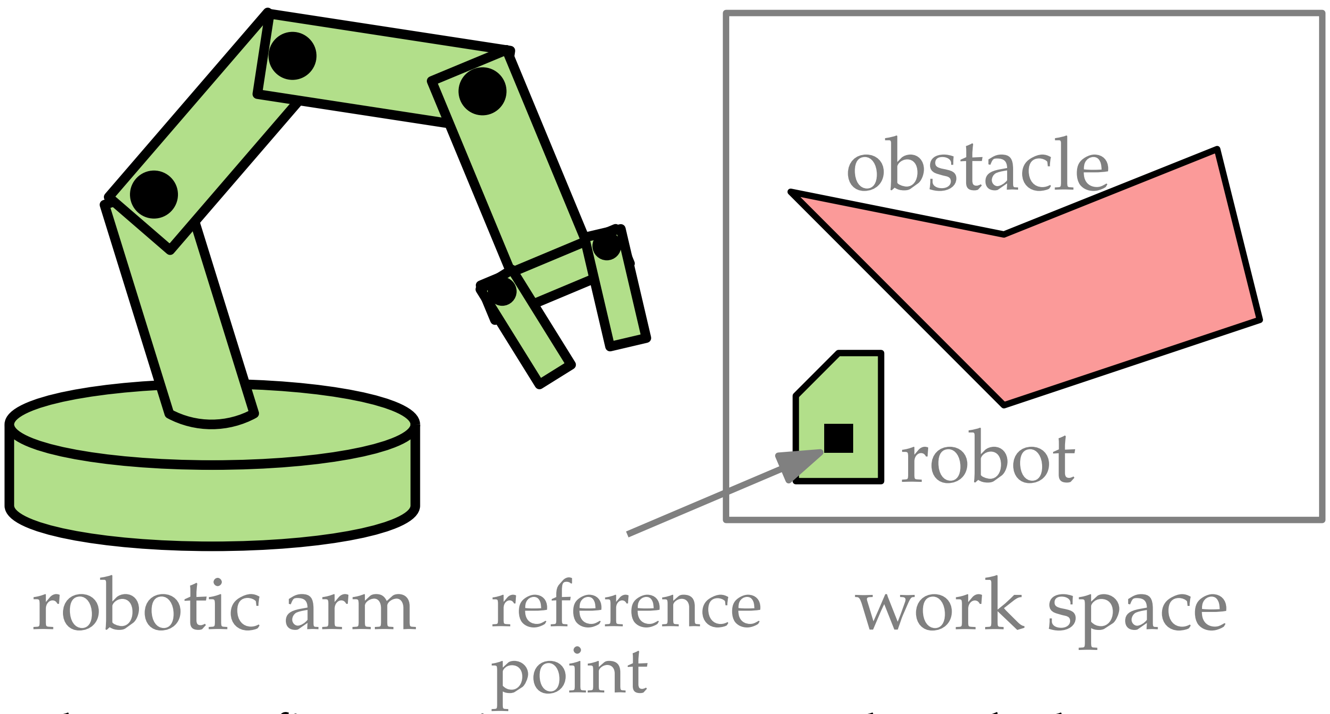
robotic arm



work space

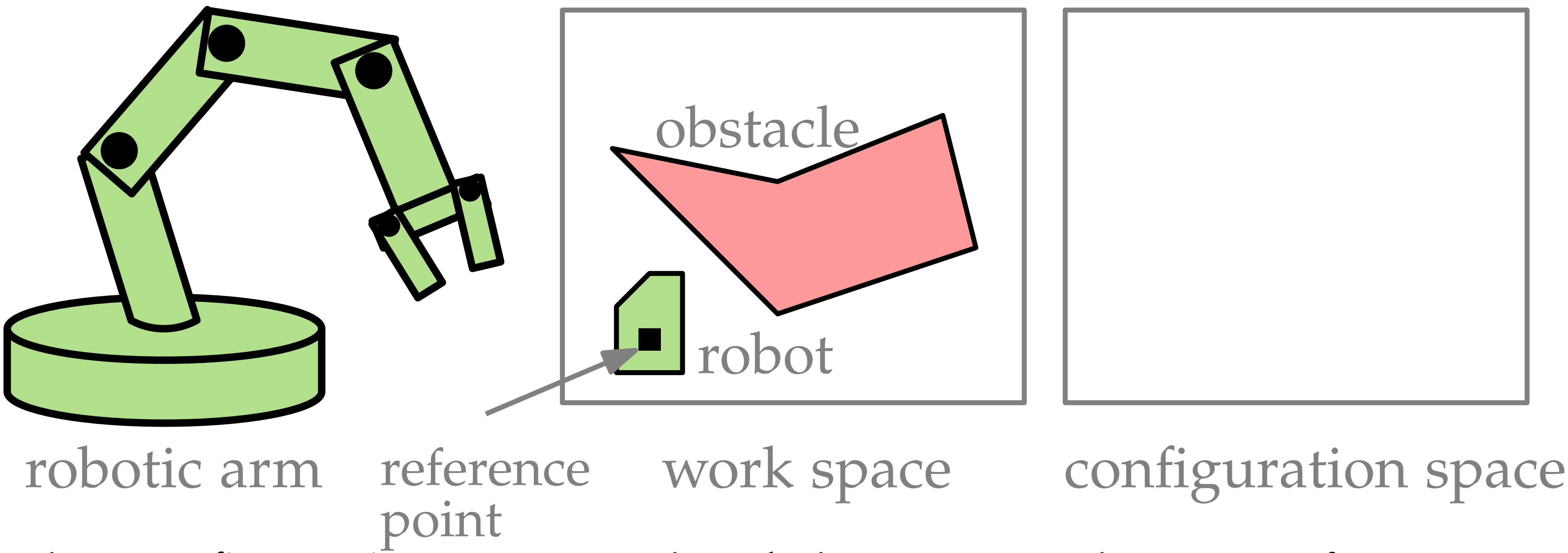
The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Configuration Space



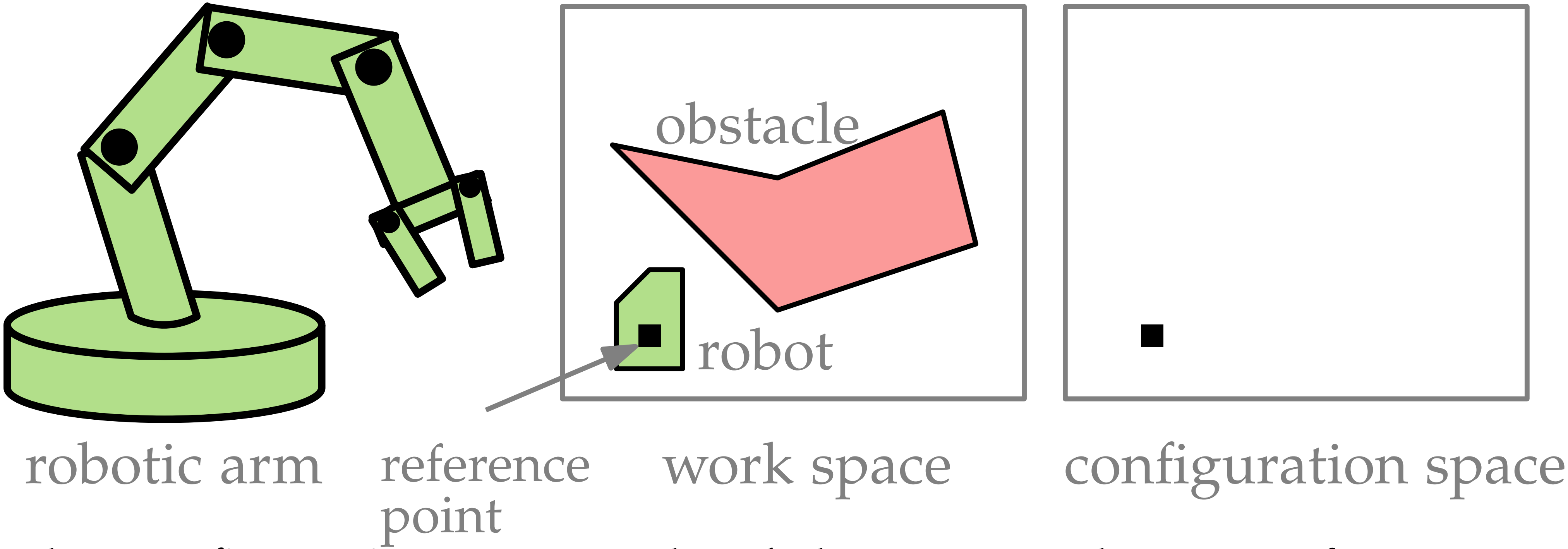
The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Configuration Space



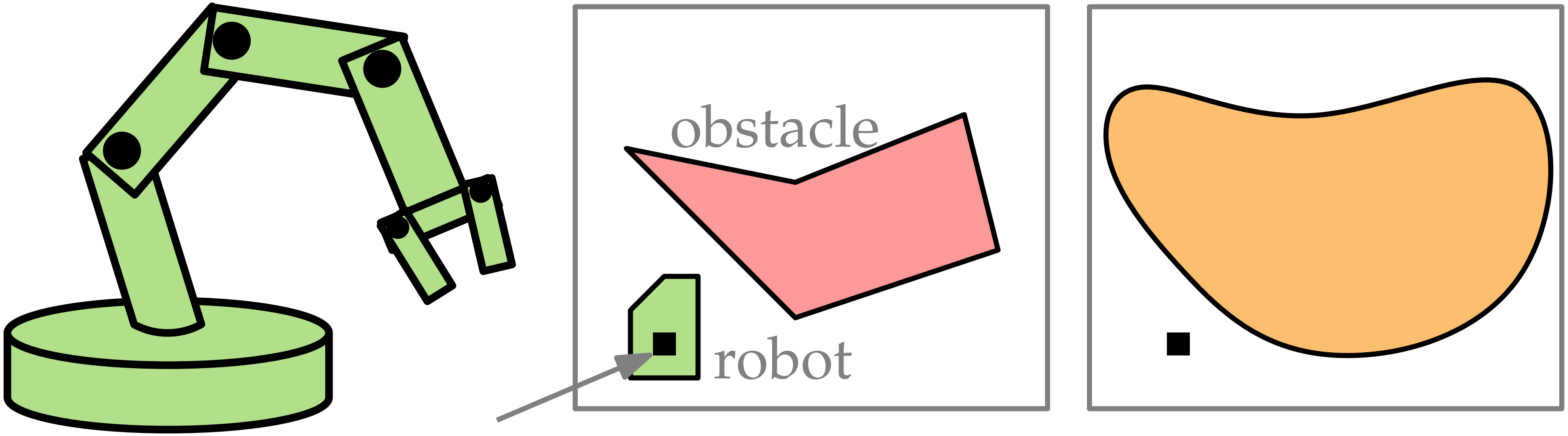
The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Configuration Space



The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

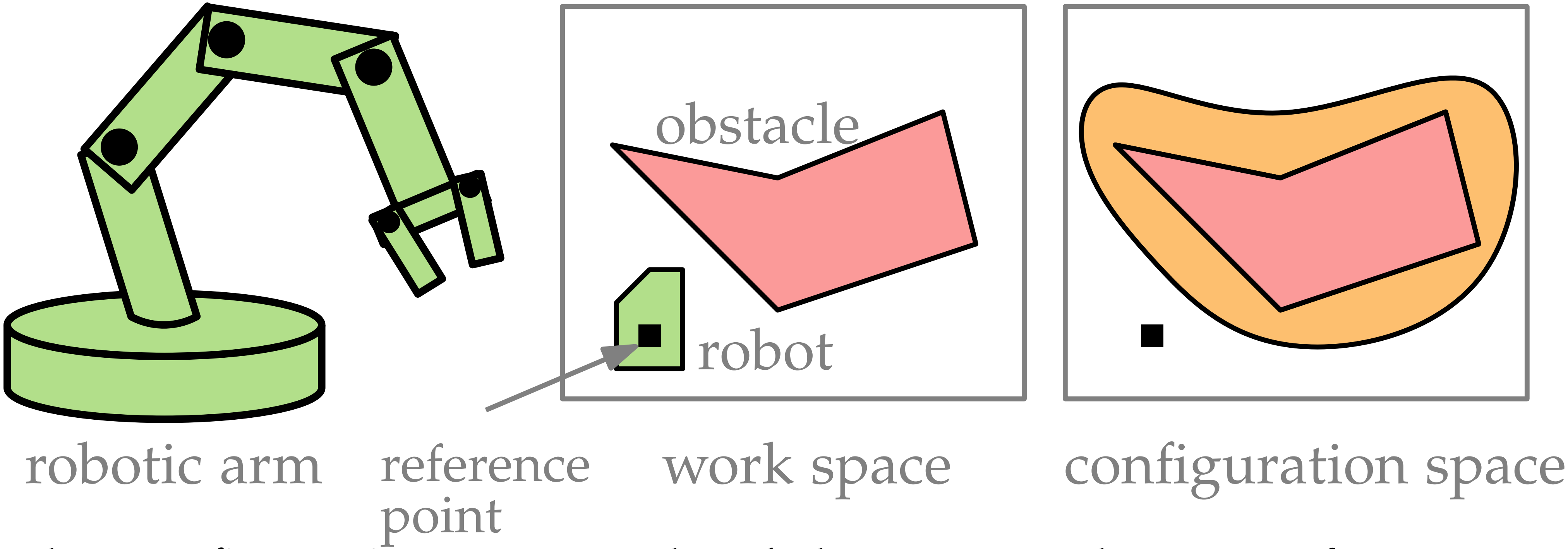
Configuration Space



robotic arm　reference point　work space　configuration space

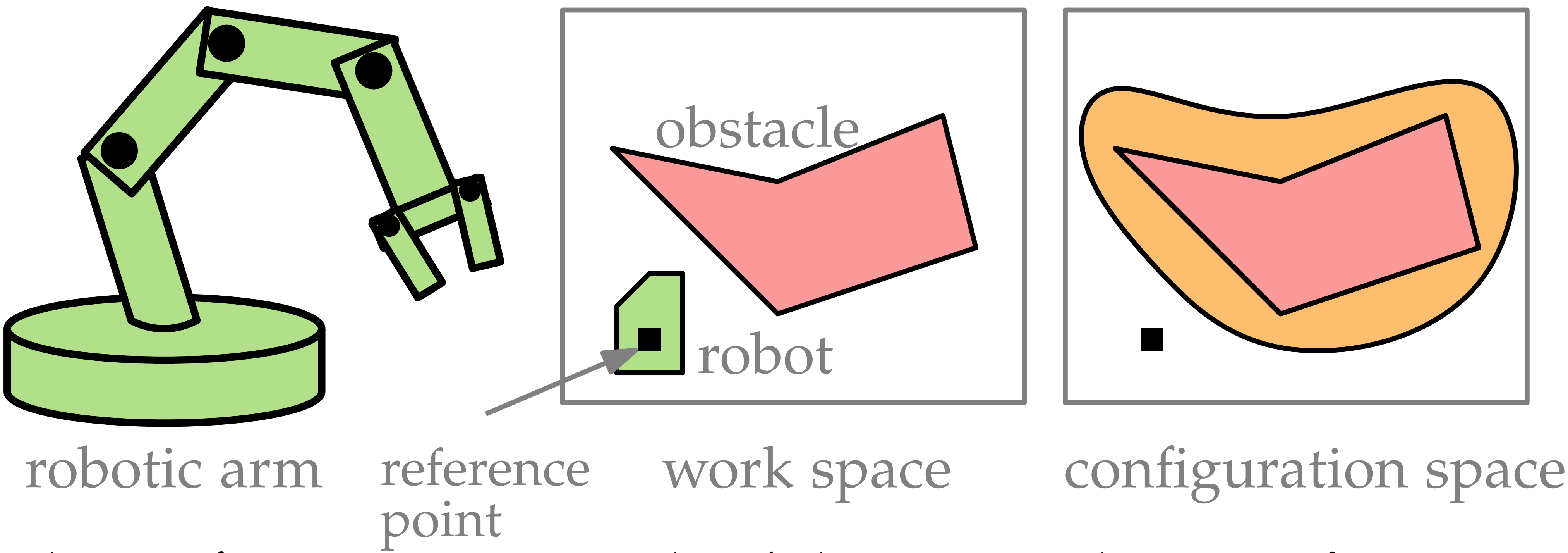
The *configuration space* is the d-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Configuration Space



The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

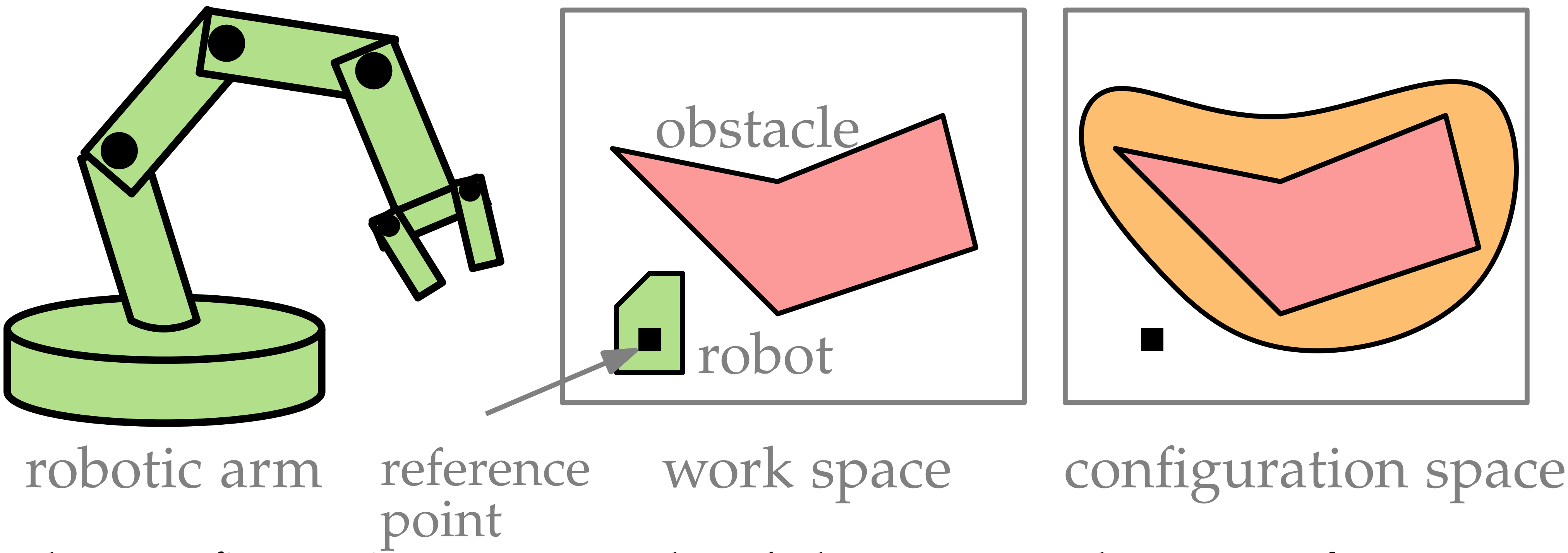
Configuration Space



The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Path for a *point* through configuration space

Configuration Space

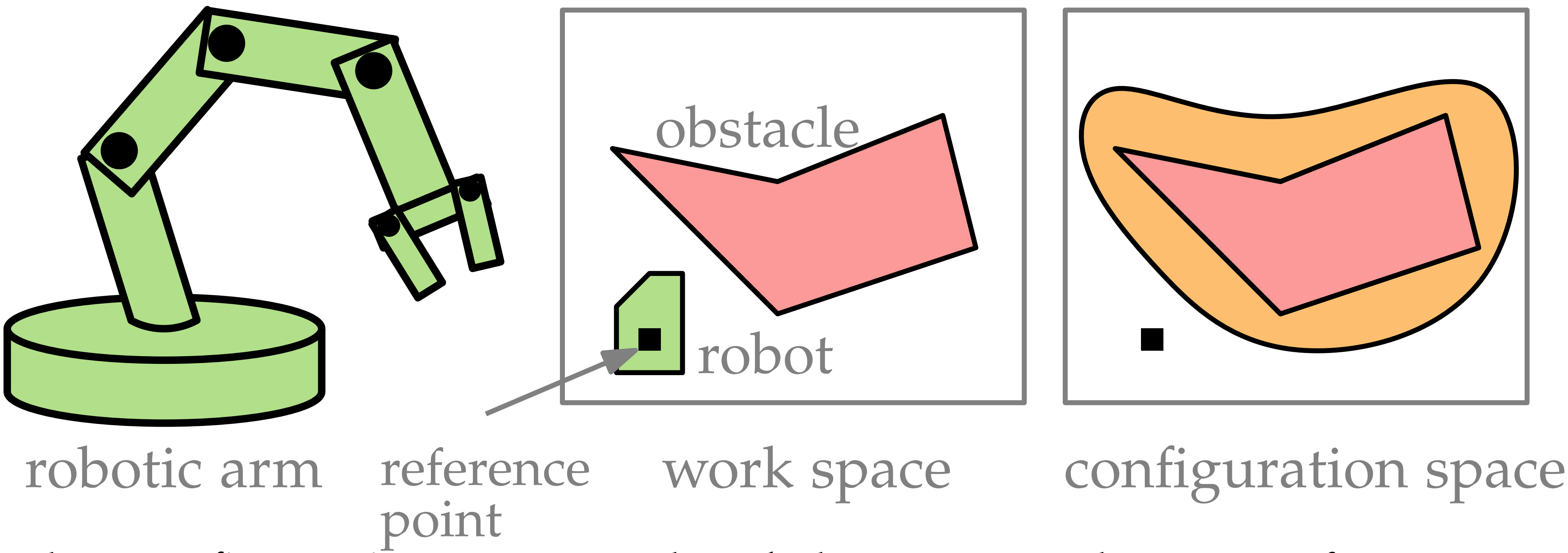


The *configuration space* is the *d*-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Path for a *point* through configuration space

$$\Downarrow$$

Configuration Space



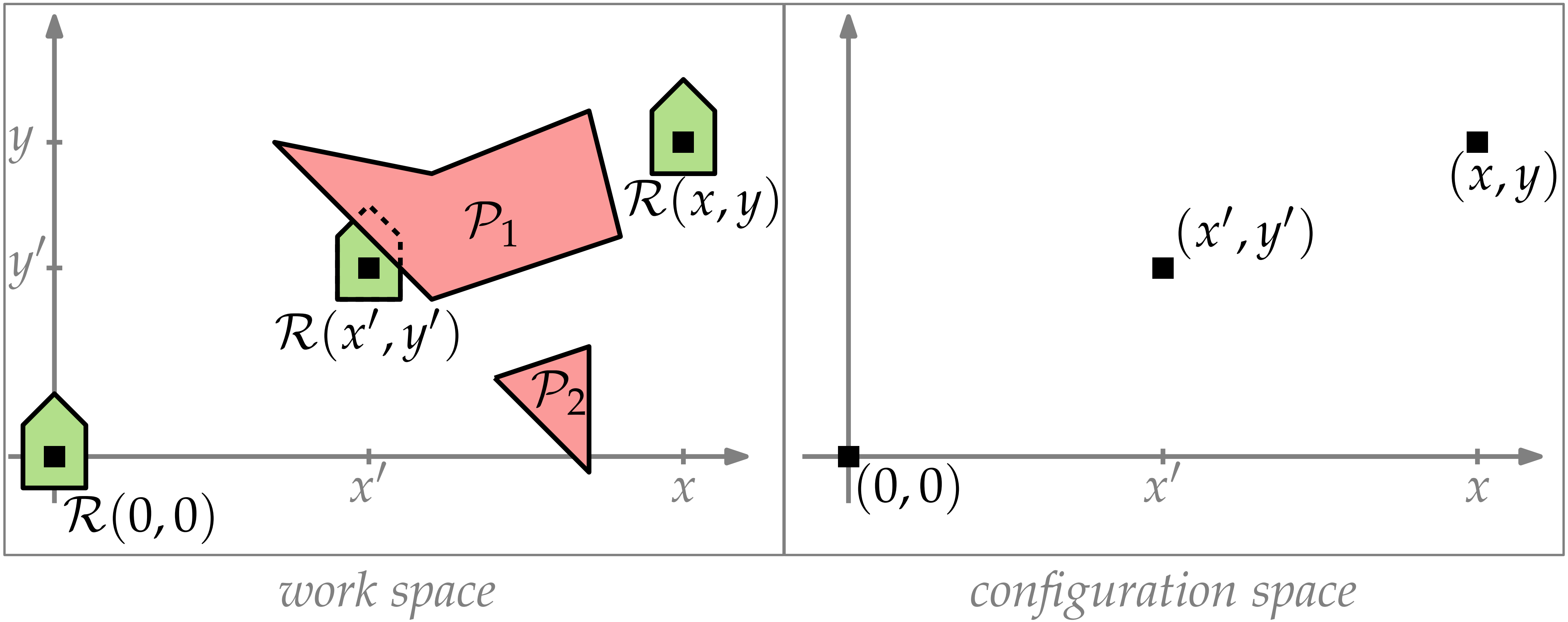
The *configuration space* is the d-dimensional space of all possible (i.e., obstacle avoiding) parameter value combinations.

Path for a *point* through configuration space

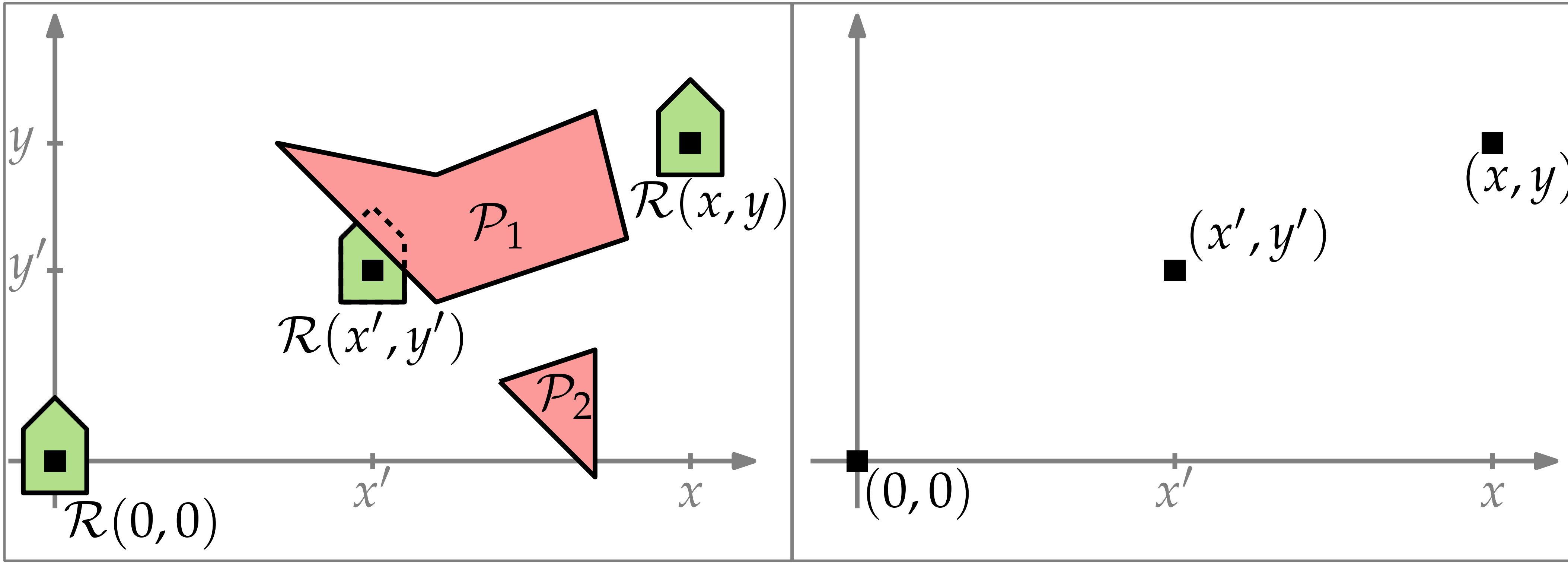
$$\Downarrow$$

path for the *robot* in the original space.

Example: Translating 2D Polygonal Robots



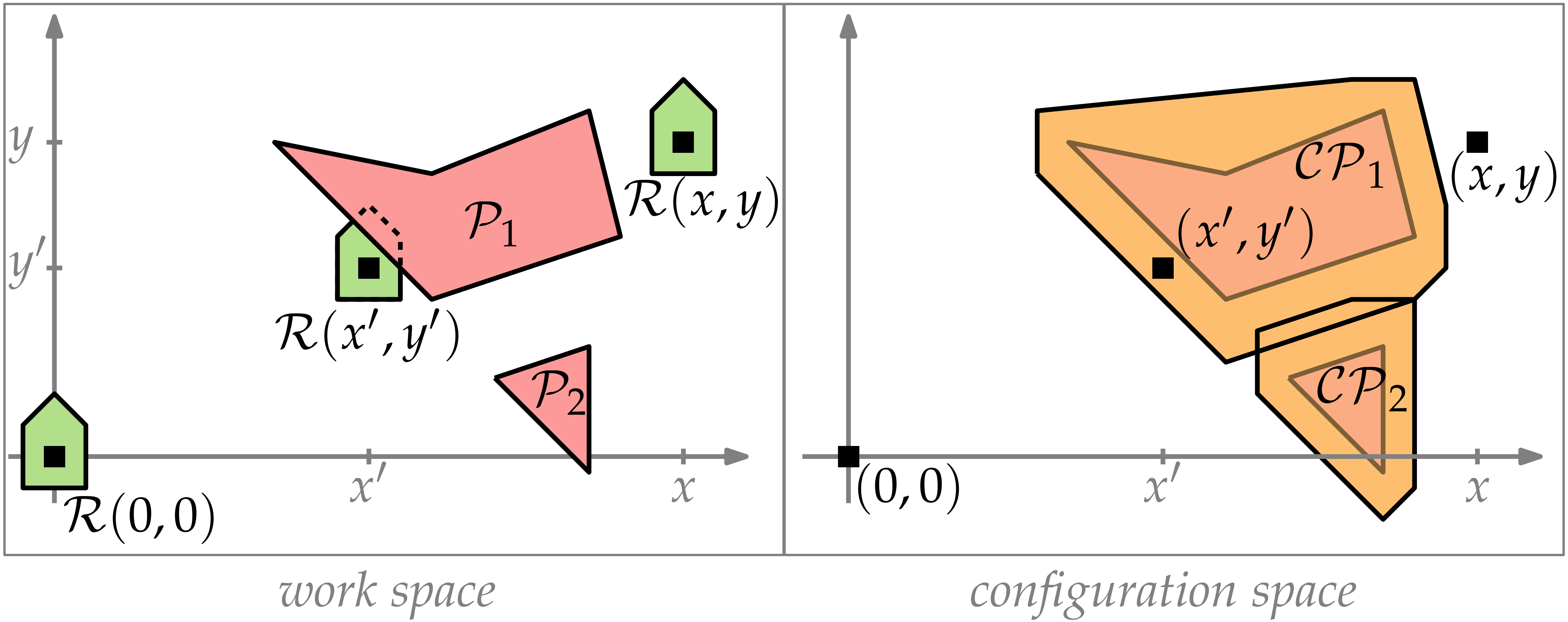
Example: Translating 2D Polygonal Robots



work space *configuration space*

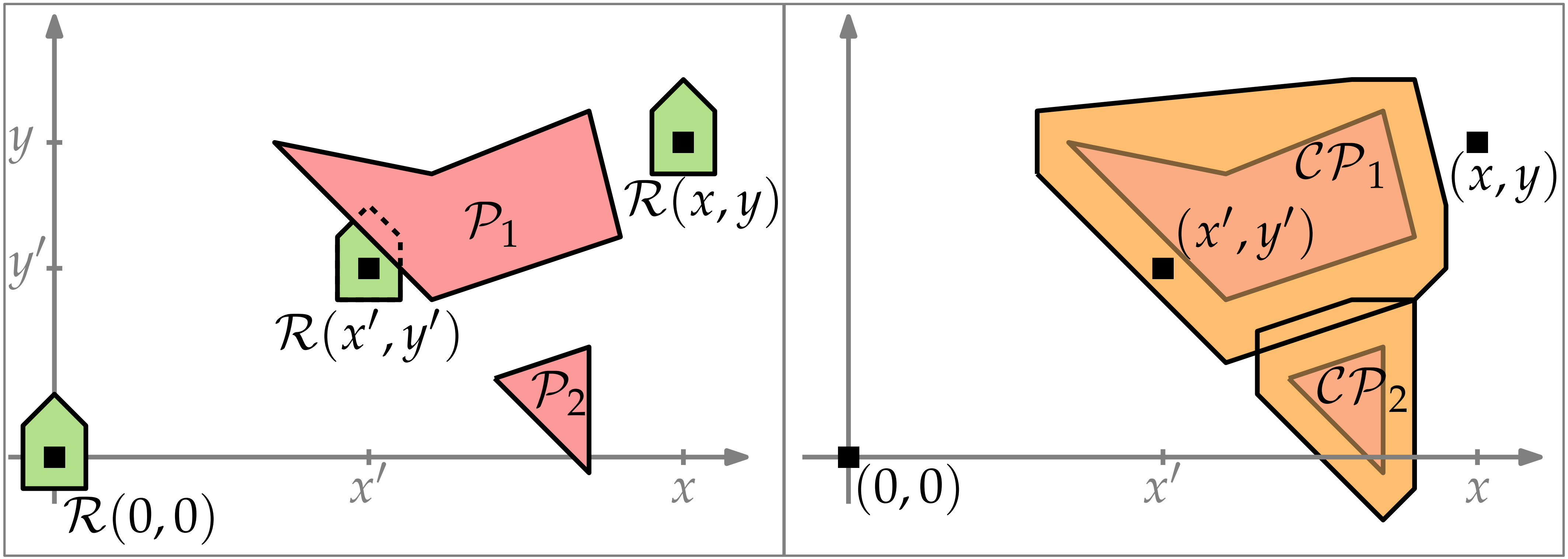
- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.

Example: Translating 2D Polygonal Robots



- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.

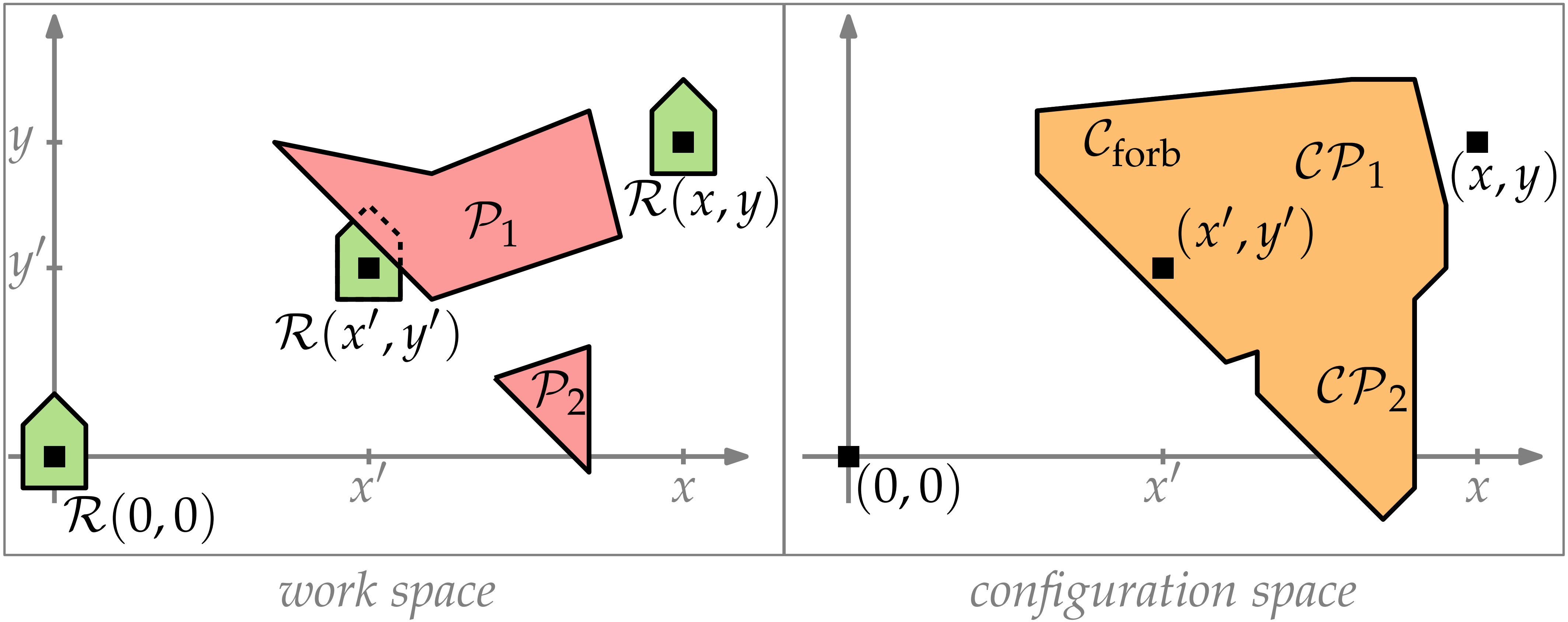
Example: Translating 2D Polygonal Robots



work space *configuration space*

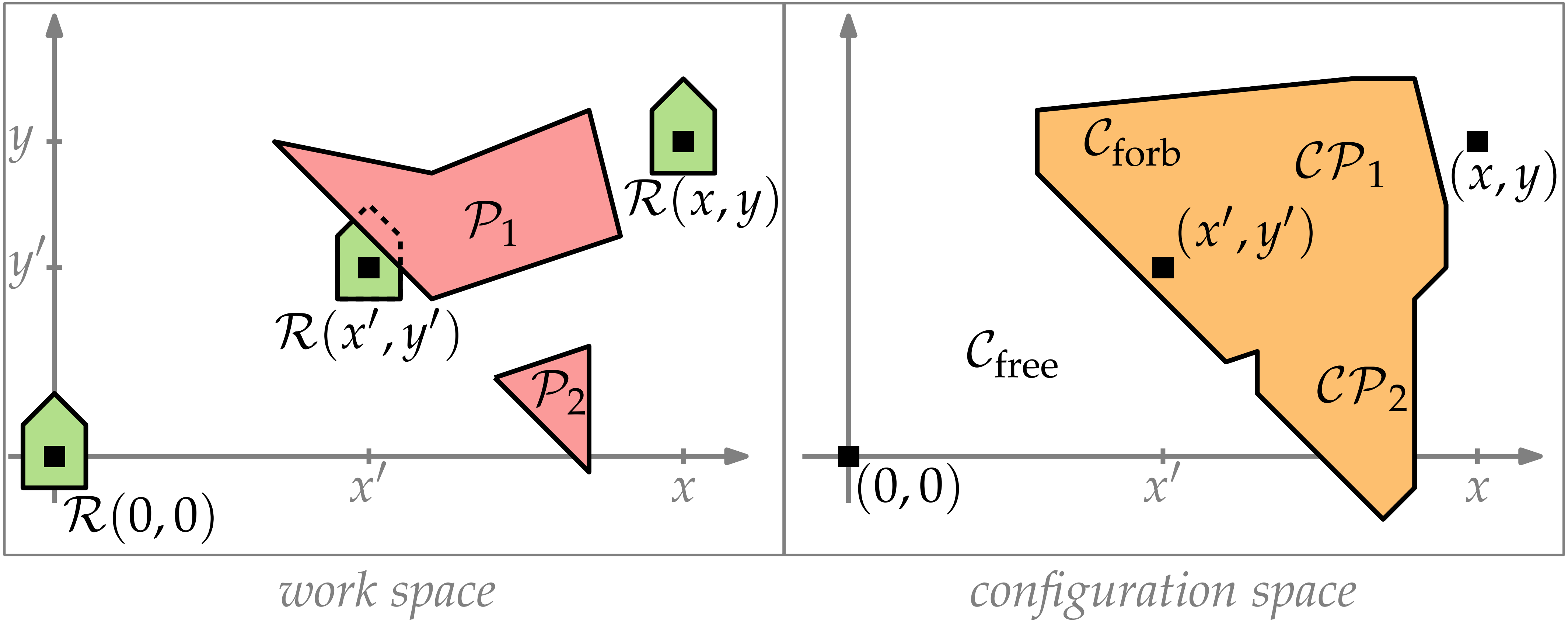
- Compute $\mathcal{CP}_i = \{(x, y) : \mathcal{R}(x, y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.

Example: Translating 2D Polygonal Robots



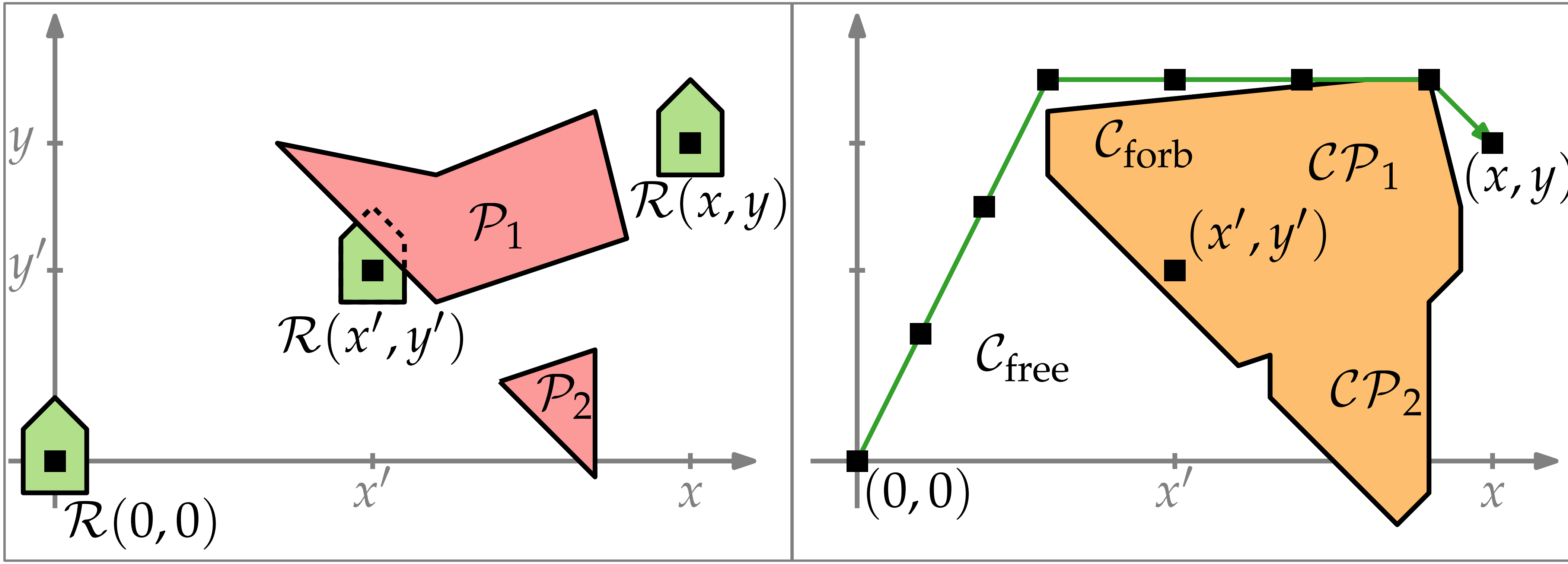
- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.

Example: Translating 2D Polygonal Robots



- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.
- Find a path for a point in the complement $\mathcal{C}_{\text{free}}$ of $\mathcal{C}_{\text{forb}}$.

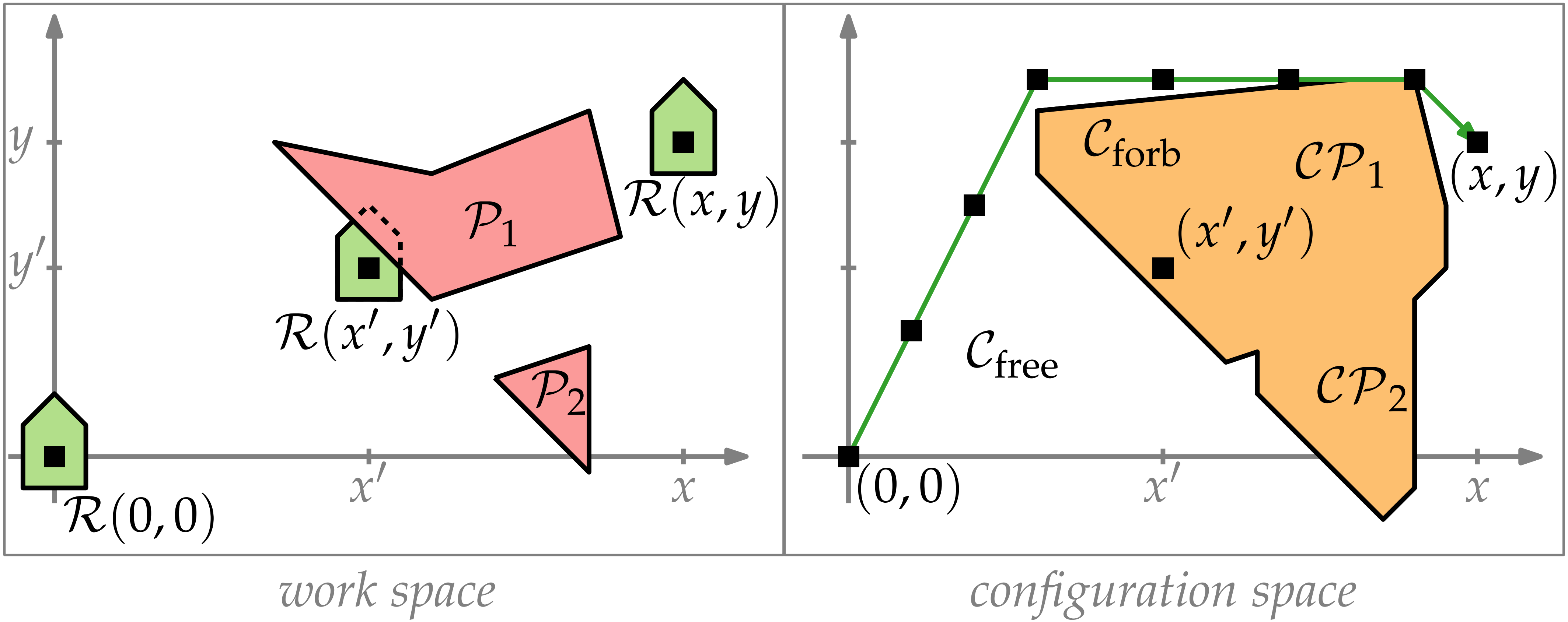
Example: Translating 2D Polygonal Robots



work space *configuration space*

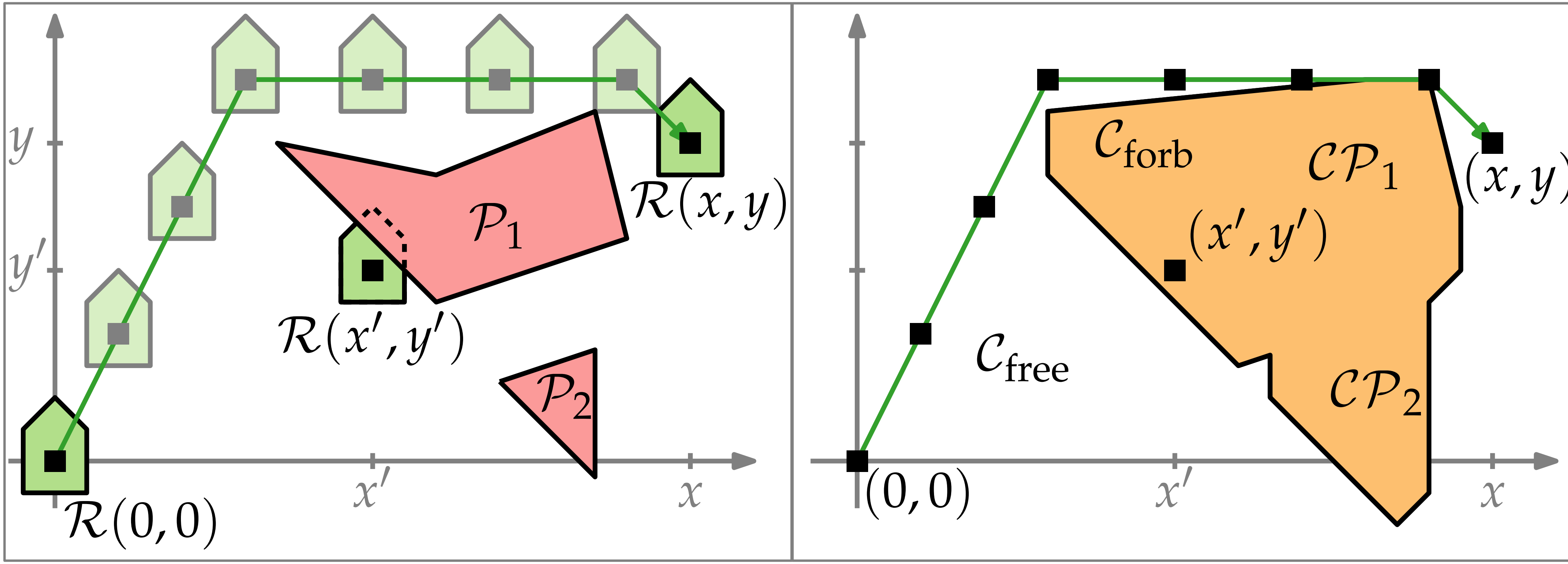
- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.
- Find a path for a point in the complement $\mathcal{C}_{\text{free}}$ of $\mathcal{C}_{\text{forb}}$.

Example: Translating 2D Polygonal Robots



- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.
- Find a path for a point in the complement $\mathcal{C}_{\text{free}}$ of $\mathcal{C}_{\text{forb}}$.
  $\Rightarrow$ collision-free path for the robot in work space

Example: Translating 2D Polygonal Robots



work space *configuration space*

- Compute $\mathcal{CP}_i = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P}_i \neq \emptyset\}$ for each $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$.
- Find a path for a point in the complement $\mathcal{C}_{\text{free}}$ of $\mathcal{C}_{\text{forb}}$.
  $\Rightarrow$ collision-free path for the robot in work space

Computational Geometry

Lecture 10:
Motion Planning

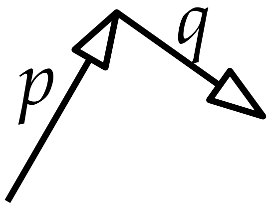
Part III:
Characterizing Configuration Spaces

Philipp Kindermann

Winter Semester 2020

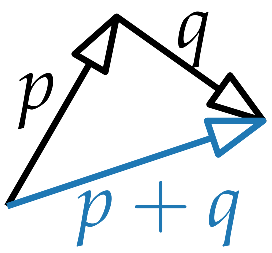
Some Linear Algebra

Vector sums



Some Linear Algebra

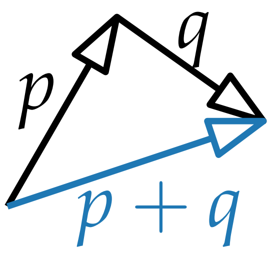
Vector sums



Some Linear Algebra

Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

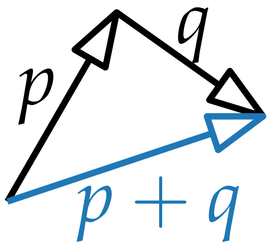


Some Linear Algebra

Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

Geometry: place vectors head to tail

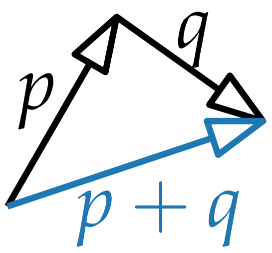


Some Linear Algebra

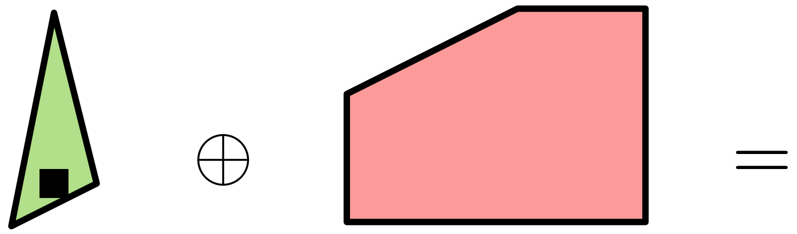
Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

Geometry: place vectors head to tail



Minkowski sums

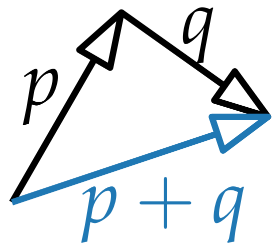


Some Linear Algebra

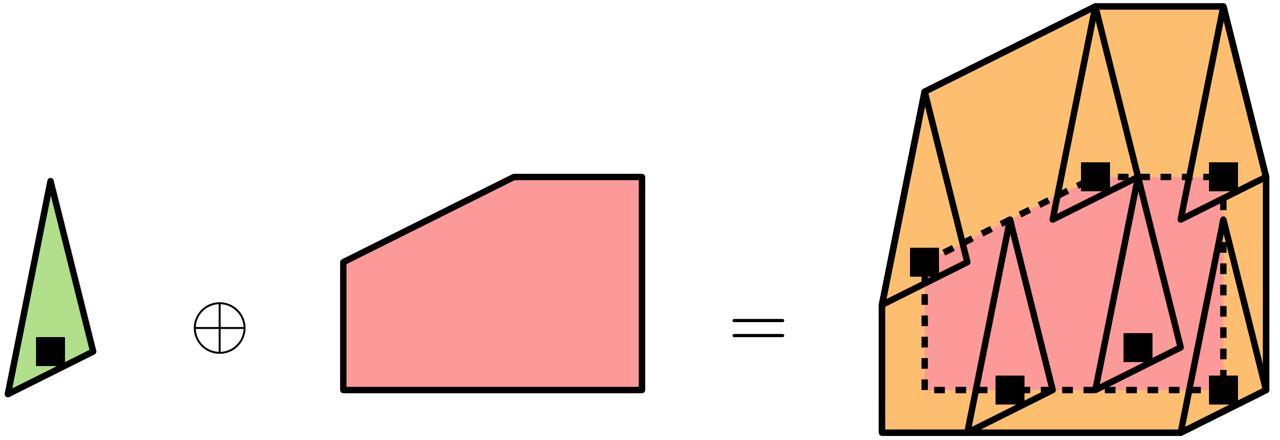
Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

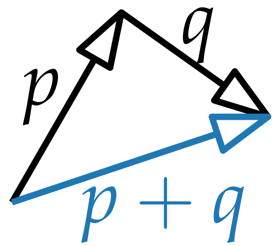
Geometry: place vectors head to tail



Minkowski sums



Some Linear Algebra



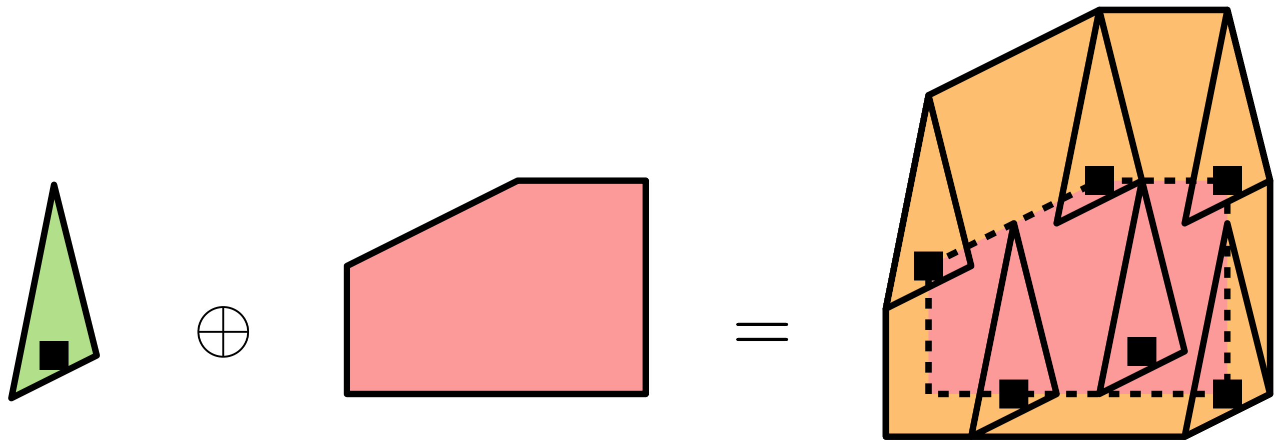
Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

Geometry: place vectors head to tail

Minkowski sums

Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

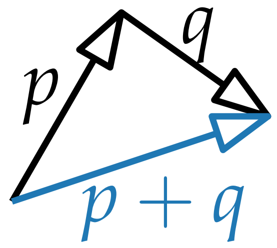


Some Linear Algebra

Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

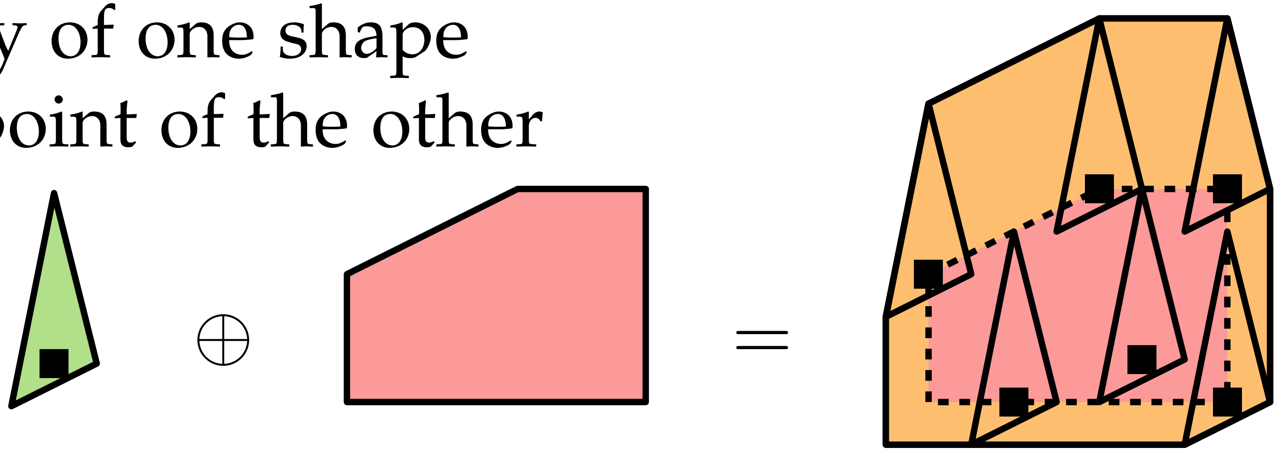
Geometry: place vectors head to tail



Minkowski sums

Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

Geometry: place copy of one shape at every point of the other

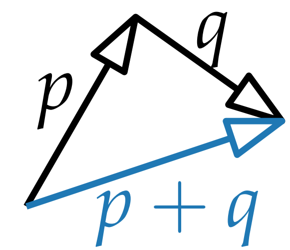


Some Linear Algebra

Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

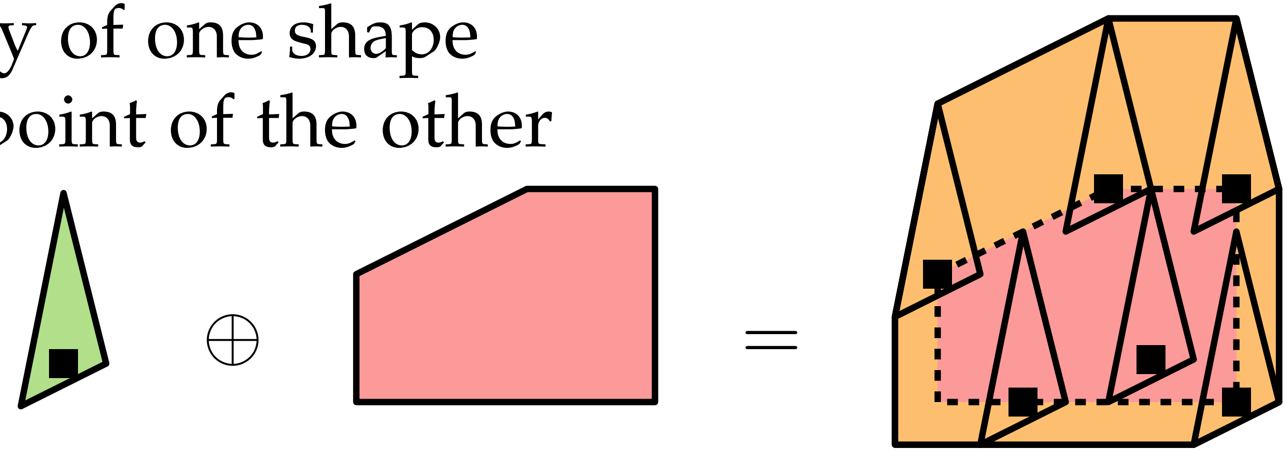
Geometry: place vectors head to tail



Minkowski sums

Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

Geometry: place copy of one shape at every point of the other



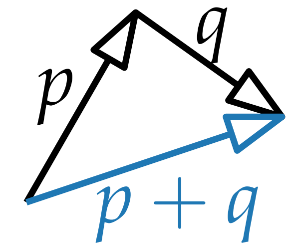
Inversion

Some Linear Algebra

Vector sums

Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

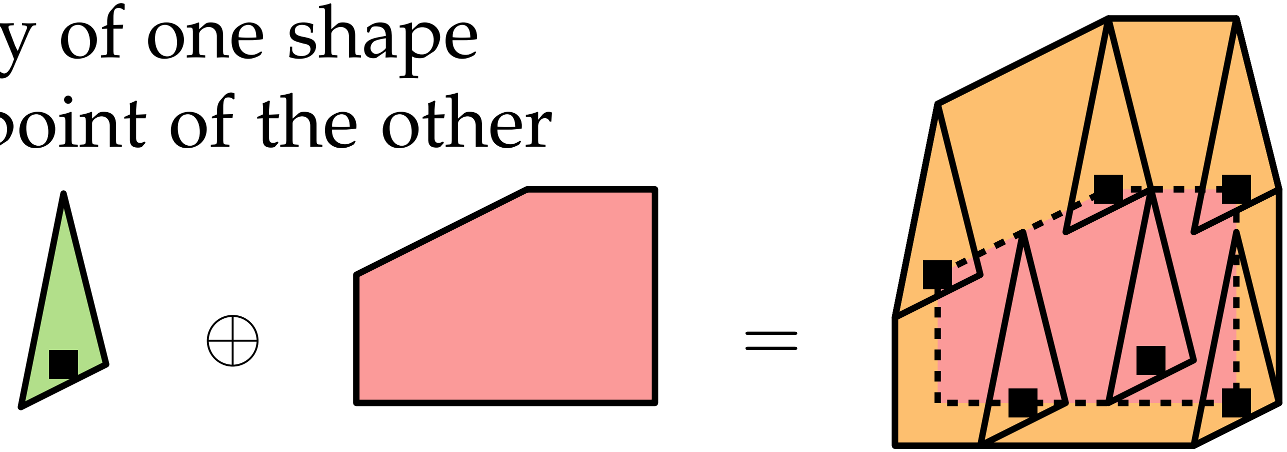
Geometry: place vectors head to tail



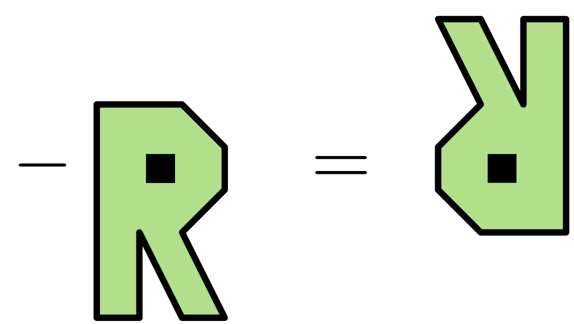
Minkowski sums

Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

Geometry: place copy of one shape at every point of the other

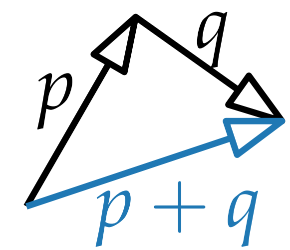


Inversion



Some Linear Algebra

Vector sums



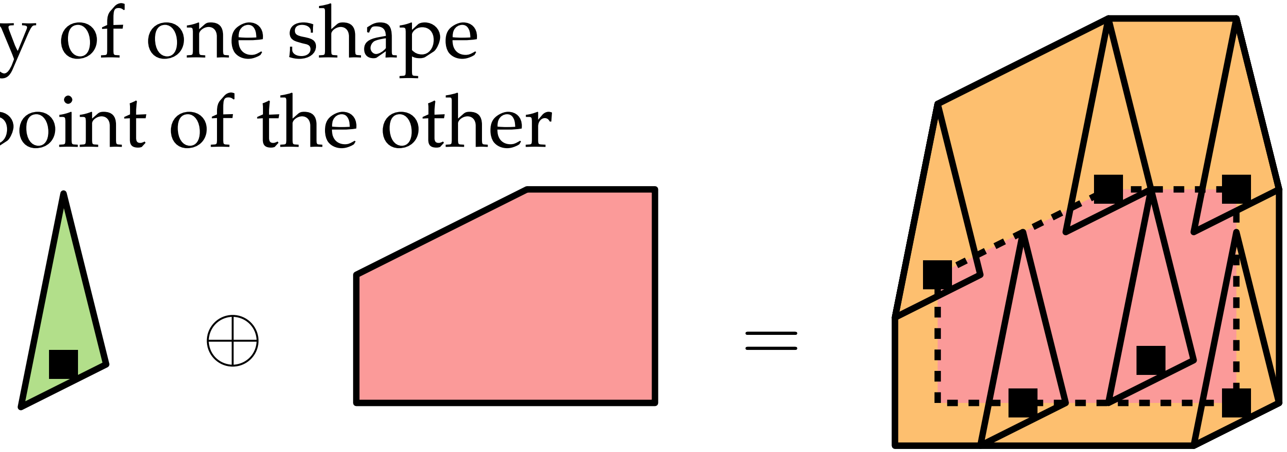
Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

Geometry: place vectors head to tail

Minkowski sums

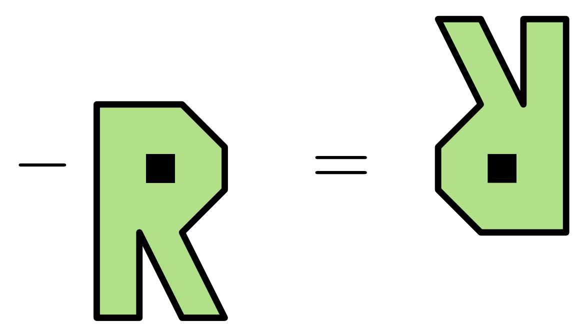
Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

Geometry: place copy of one shape at every point of the other



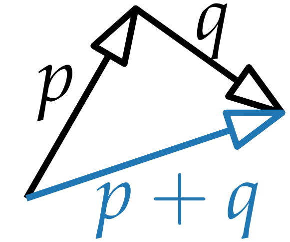
Inversion

Algebra: $-S = \{-p \mid p \in S\}$



Some Linear Algebra

Vector sums



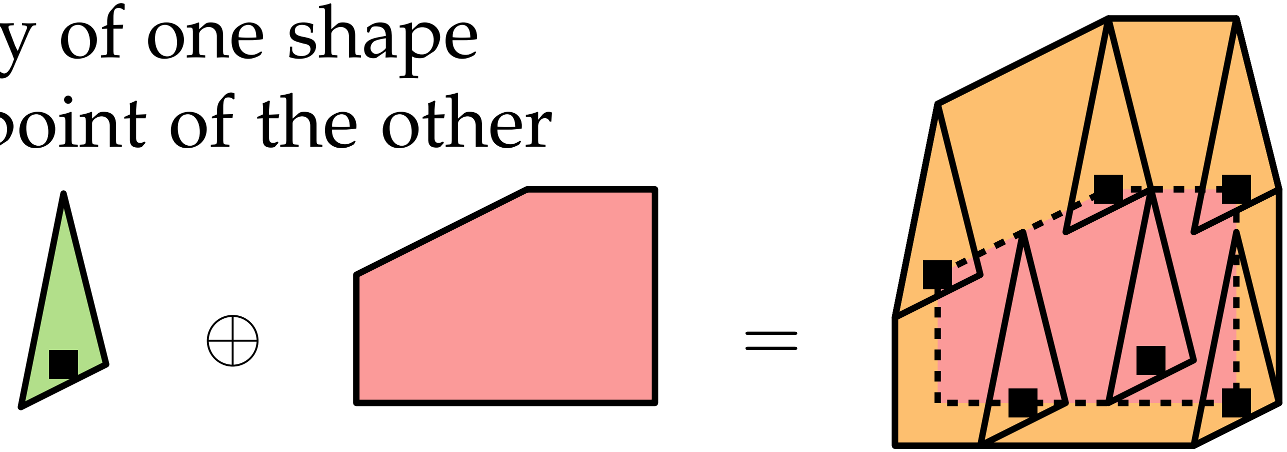
Algebra: $(p_x, p_y) + (q_x, q_y) = (p_x + q_x, p_y + q_y)$

Geometry: place vectors head to tail

Minkowski sums

Algebra: $S_1 \oplus S_2 = \{p + q \mid p \in S_1, q \in S_2\}$

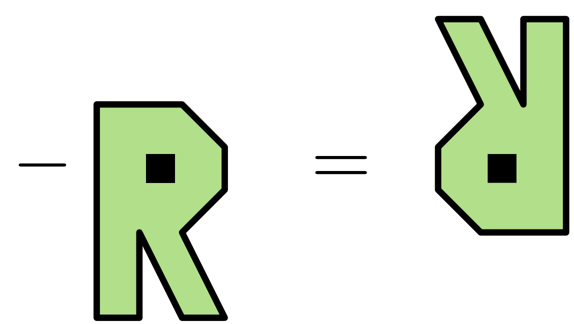
Geometry: place copy of one shape
at every point of the other



Inversion

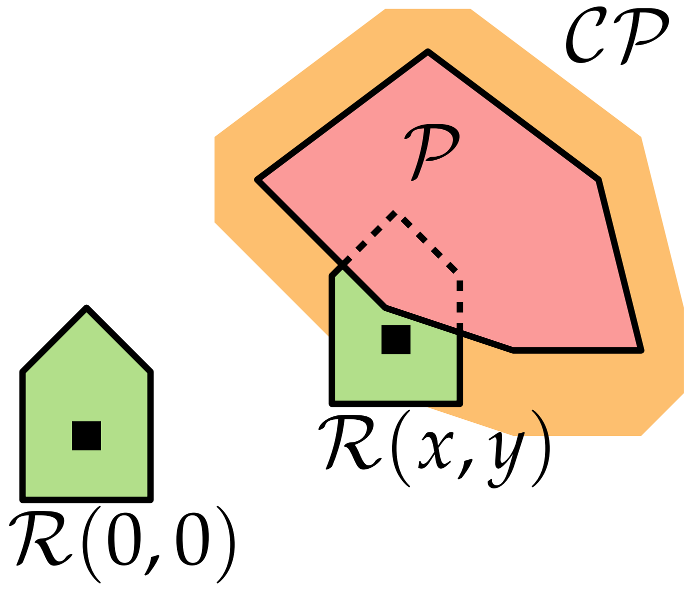
Algebra: $-S = \{-p \mid p \in S\}$

Geometry: rotate 180° (point-mirror)
around reference point



Characterizing $\mathcal{CP}$

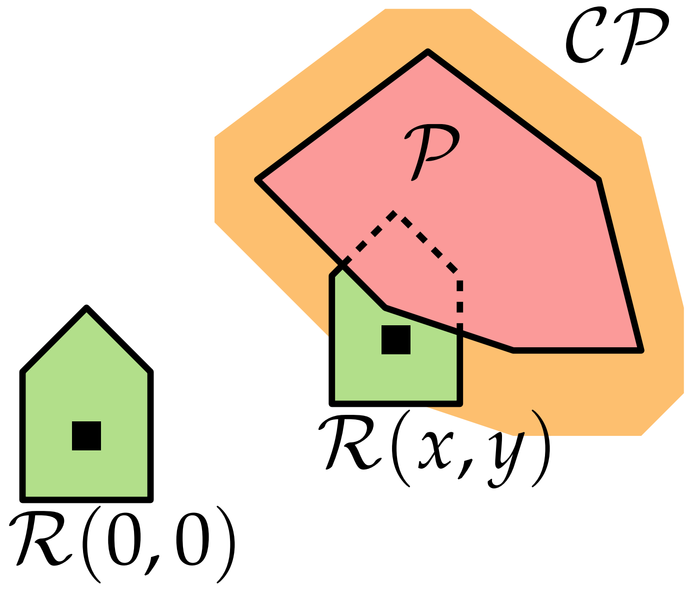
Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

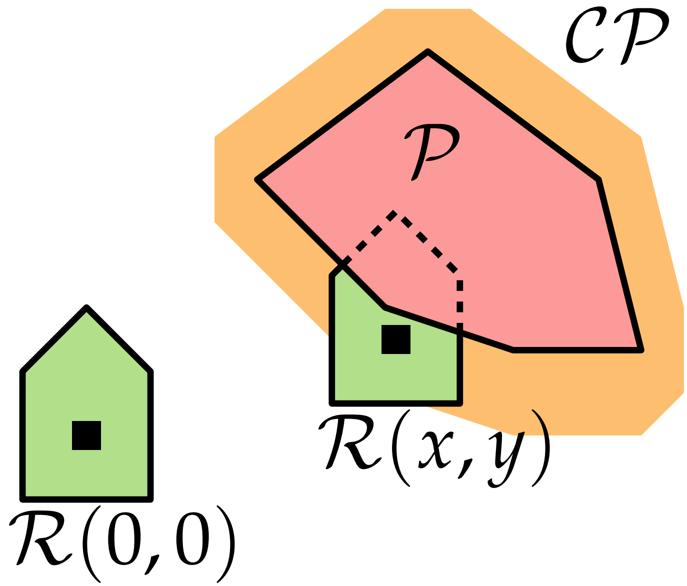


Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x, y) : \mathcal{R}(x, y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

In other words: $\mathcal{R}(x, y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x, y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0, 0))$



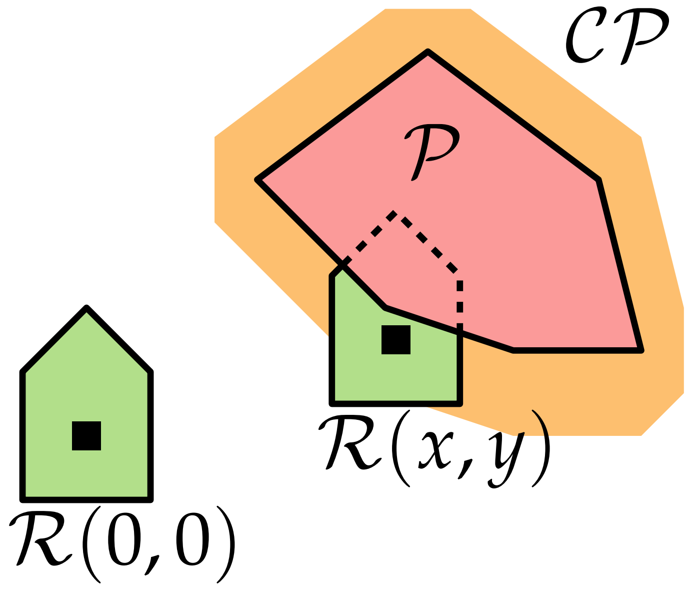
Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof.



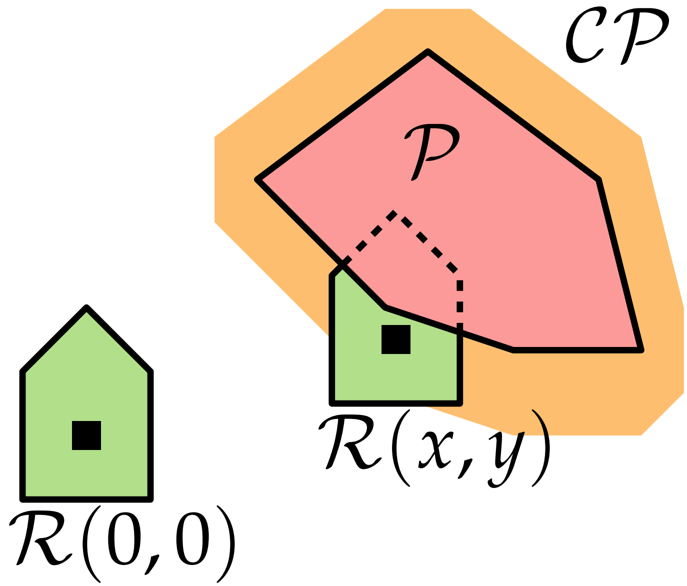
Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

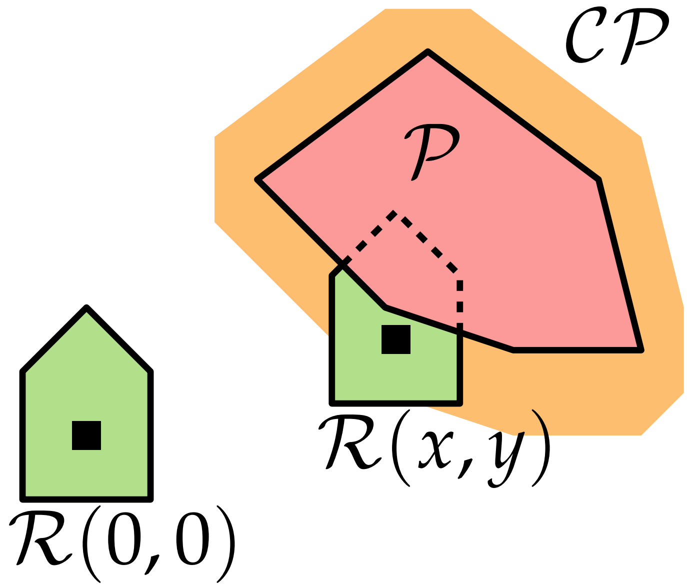
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$"

"$\Leftarrow$"



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

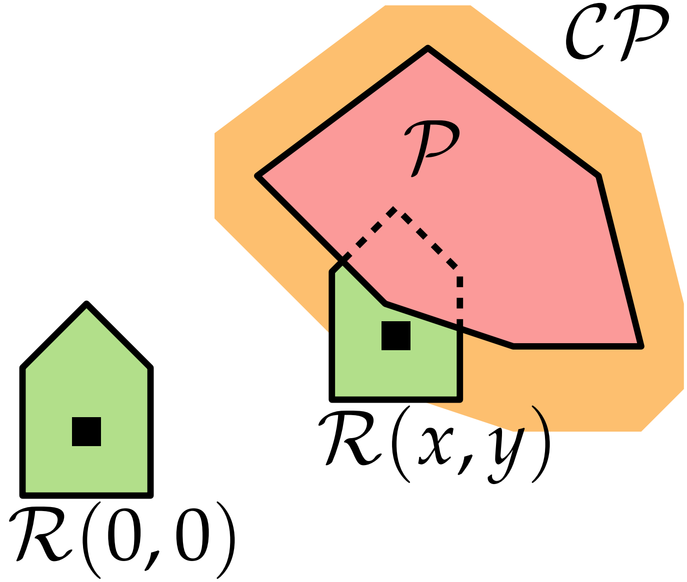
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.

"$\Leftarrow$"



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

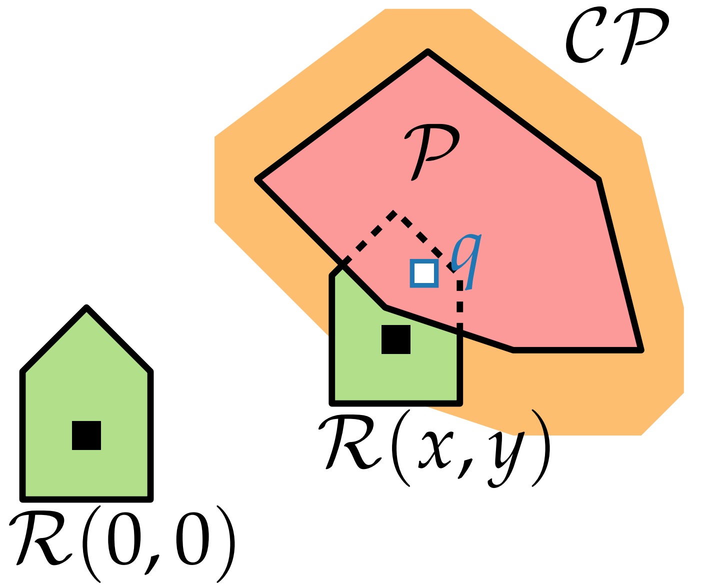
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$"



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

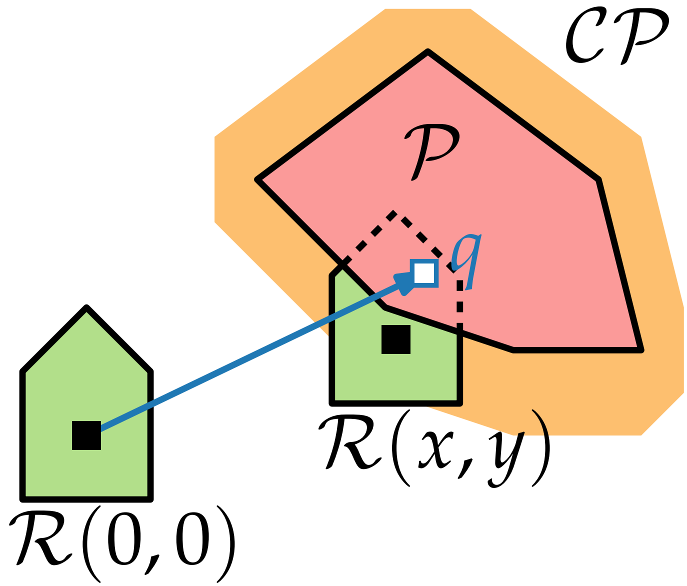
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$"



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

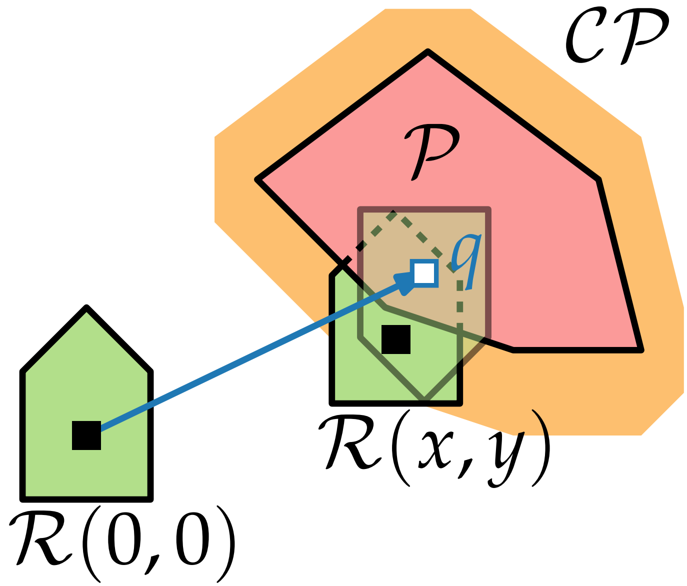
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$"



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) : \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

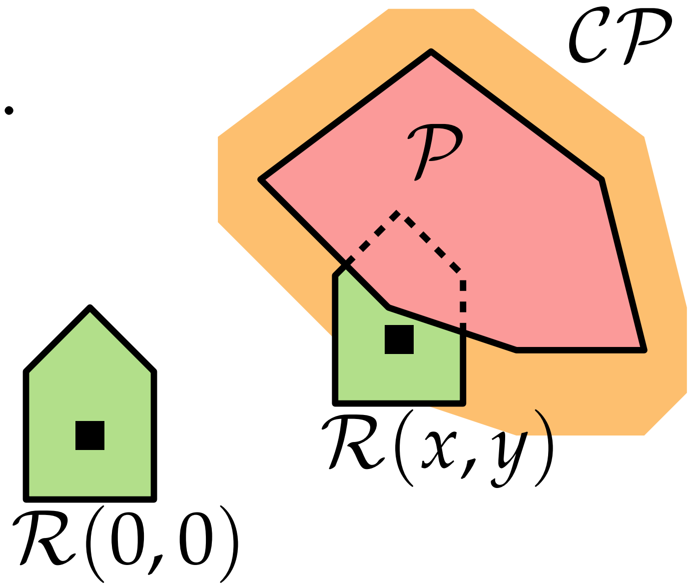
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

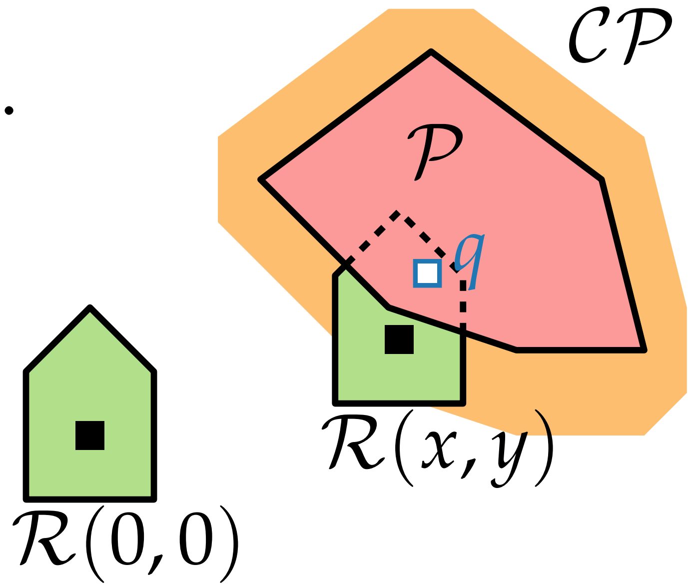
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.
Then there are points
$q \in \mathcal{P}$ and $r \in \mathcal{R}(0,0)$
such that ...



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

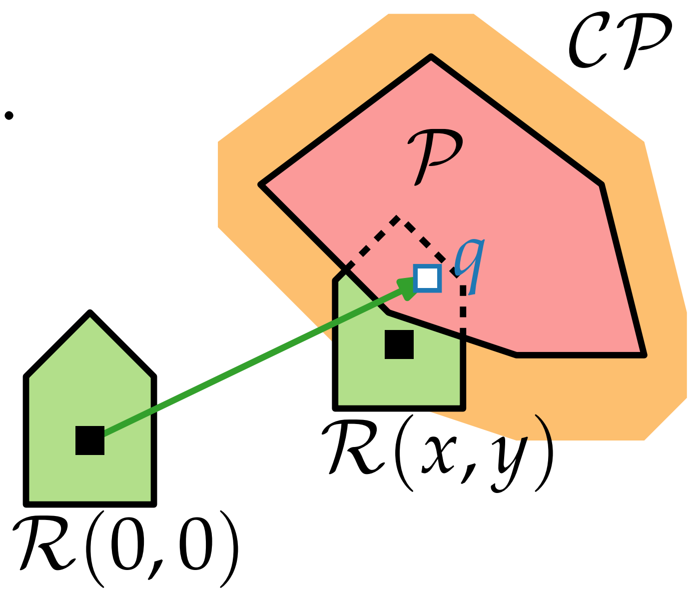
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.
Then there are points
$q \in \mathcal{P}$ and $r \in \mathcal{R}(0,0)$
such that ...



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

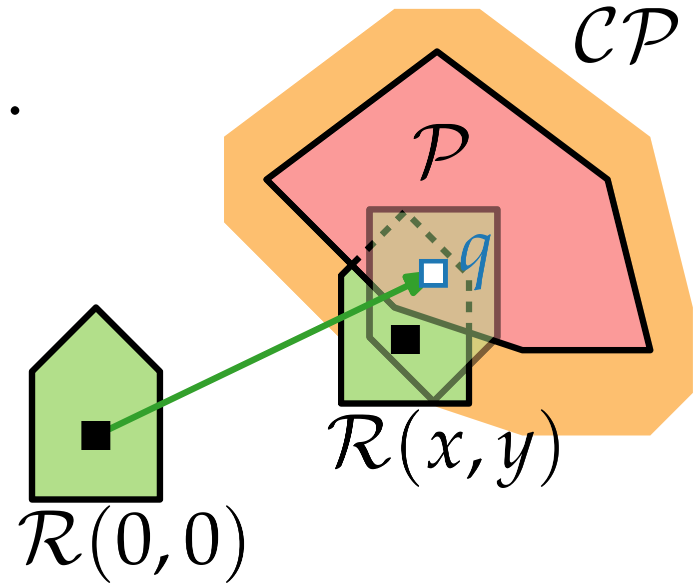
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.
Then there are points $q \in \mathcal{P}$ and $r \in \mathcal{R}(0,0)$ such that . . .



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

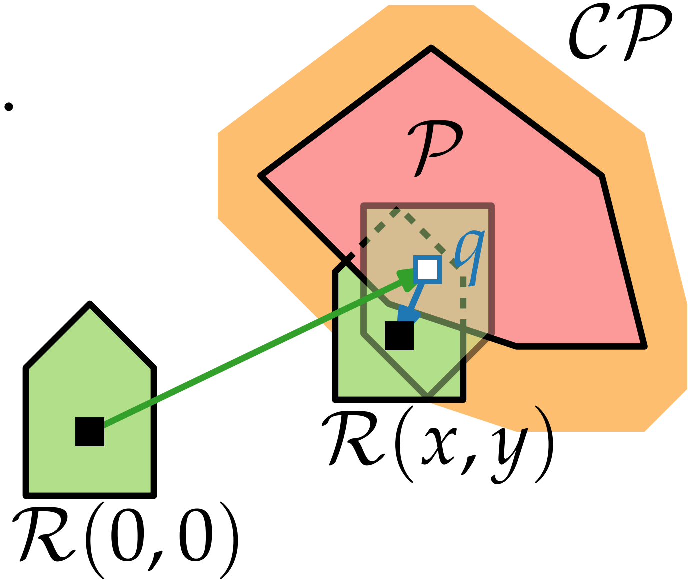
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.
Then there are points
$q \in \mathcal{P}$ and $r \in \mathcal{R}(0,0)$
such that ...



Characterizing $\mathcal{CP}$

Recall that $\mathcal{CP} = \{(x,y) \colon \mathcal{R}(x,y) \cap \mathcal{P} \neq \emptyset\}$ for an obstacle $\mathcal{P}$.

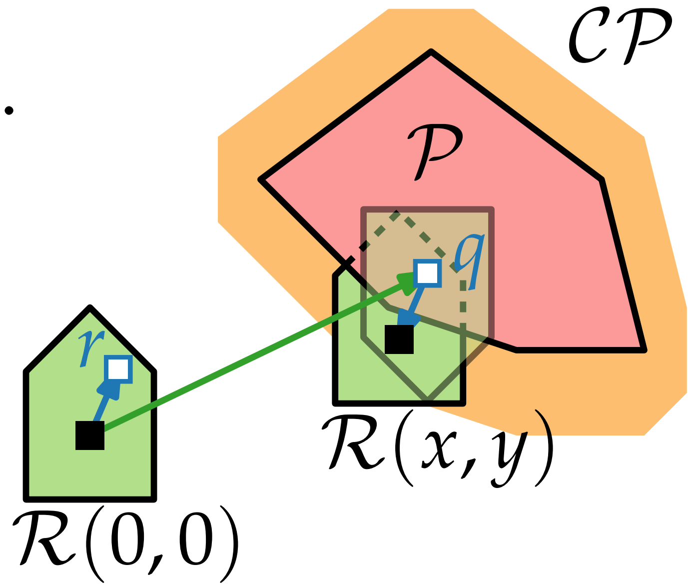
In other words: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \quad \Leftrightarrow \quad (x,y) \in \mathcal{CP}$.

Theorem. $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Proof. Show: $\mathcal{R}(x,y)$ intersects $\mathcal{P} \Leftrightarrow (x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.

"$\Rightarrow$" Suppose $\mathcal{R}(x,y)$ intersects $\mathcal{P}$.
Let $q \in \mathcal{R}(x,y) \cap \mathcal{P}$. Then...

"$\Leftarrow$" Let $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$.
Then there are points
$q \in \mathcal{P}$ and $r \in \mathcal{R}(0,0)$
such that ...



Computational Geometry

Lecture 10:
Motion Planning

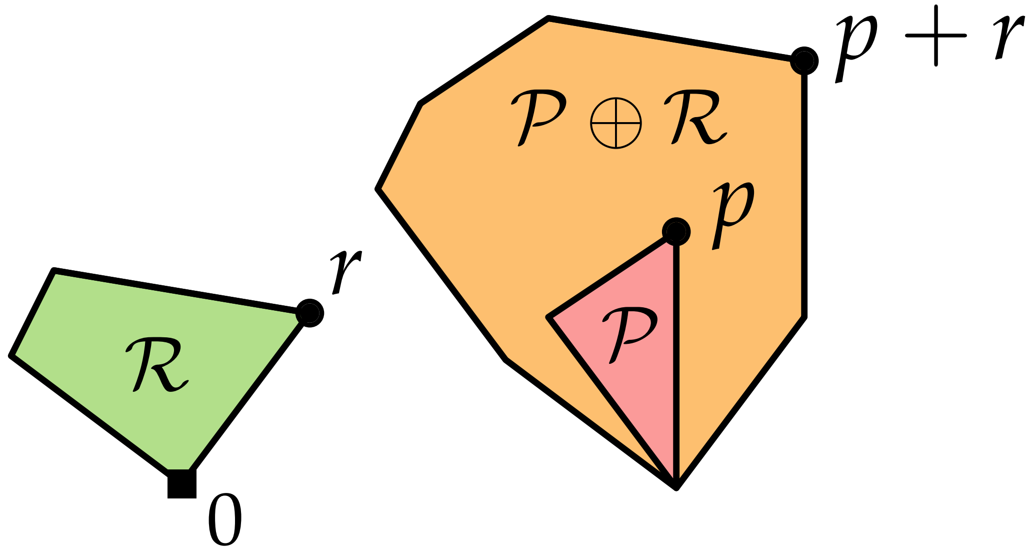
Part IV:
Minkowski Sum: Complexity & Computation

Philipp Kindermann

Winter Semester 2020

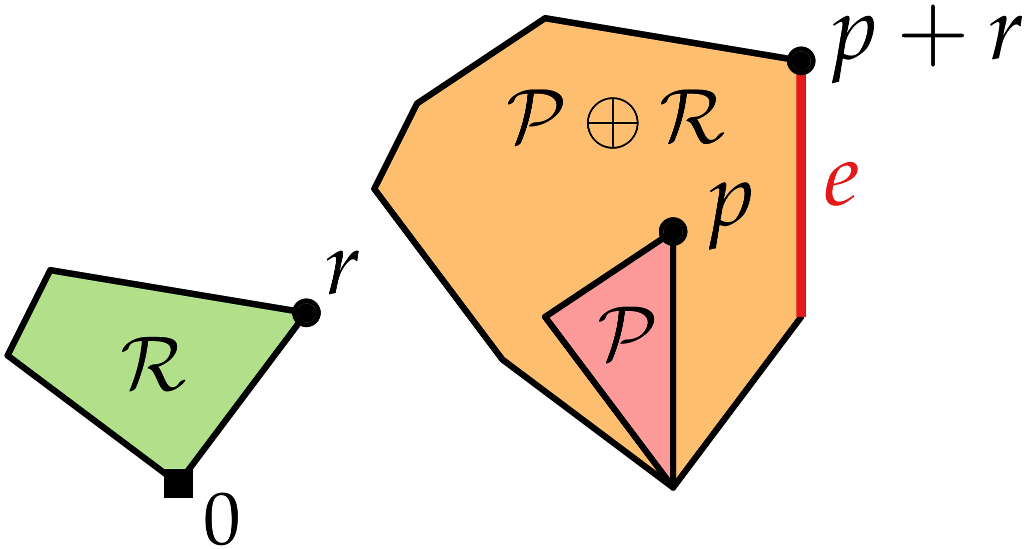
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



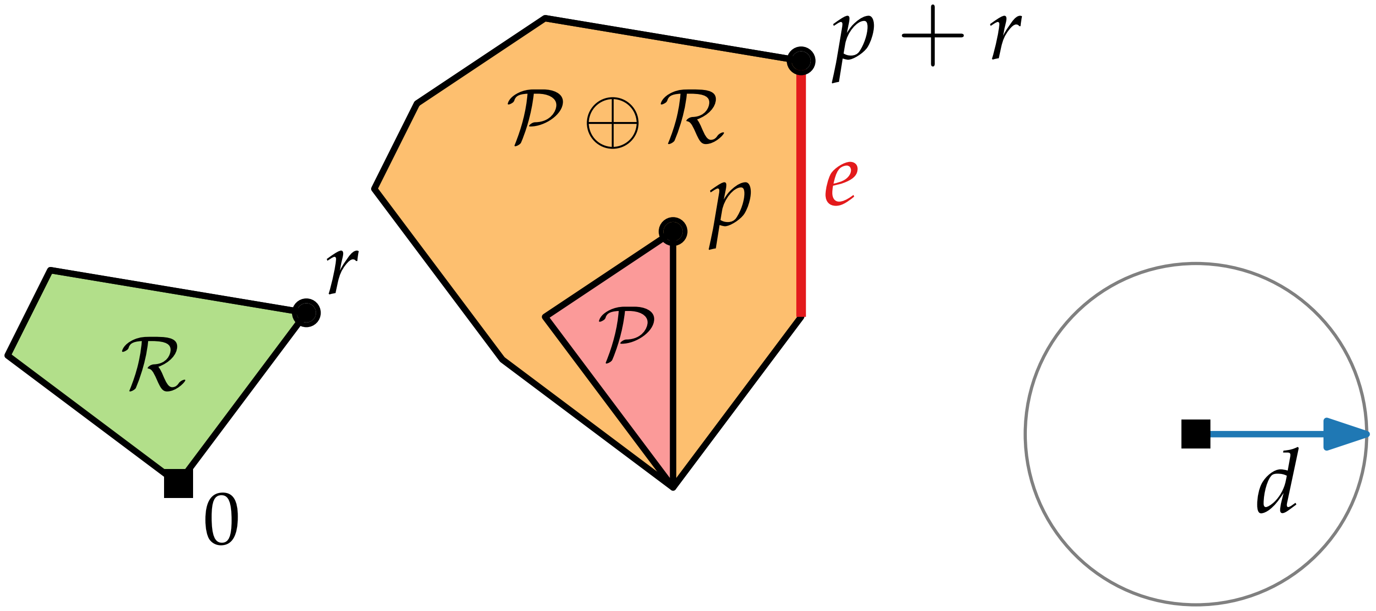
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



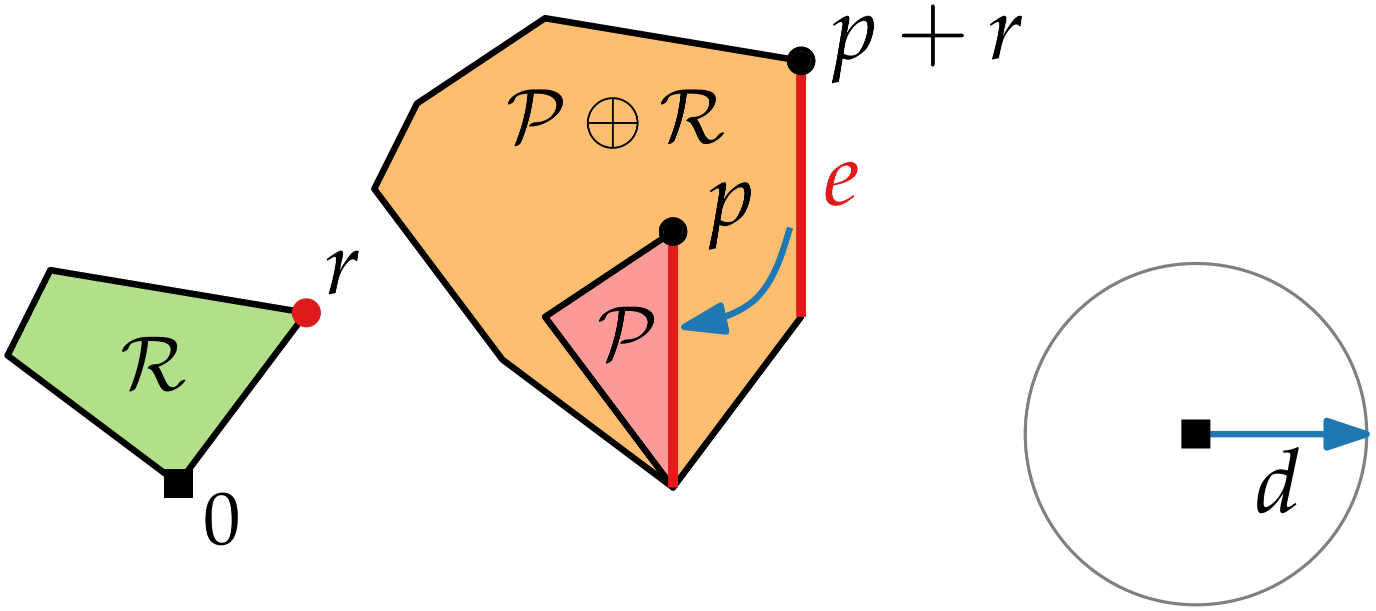
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



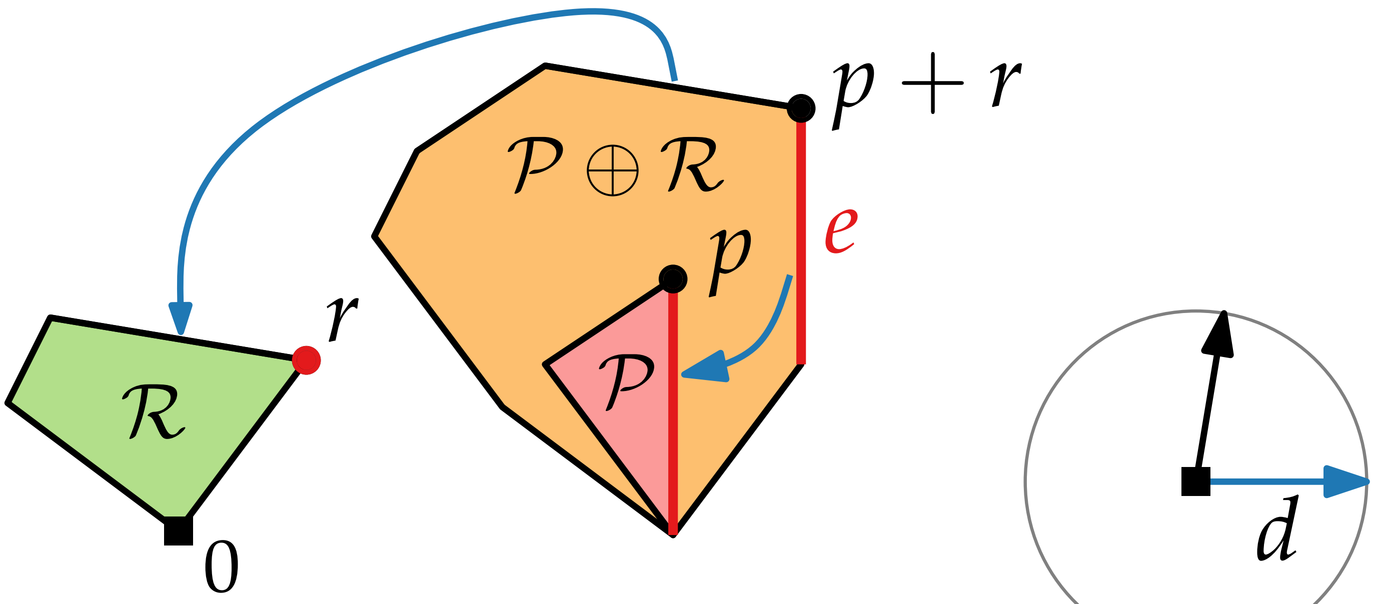
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



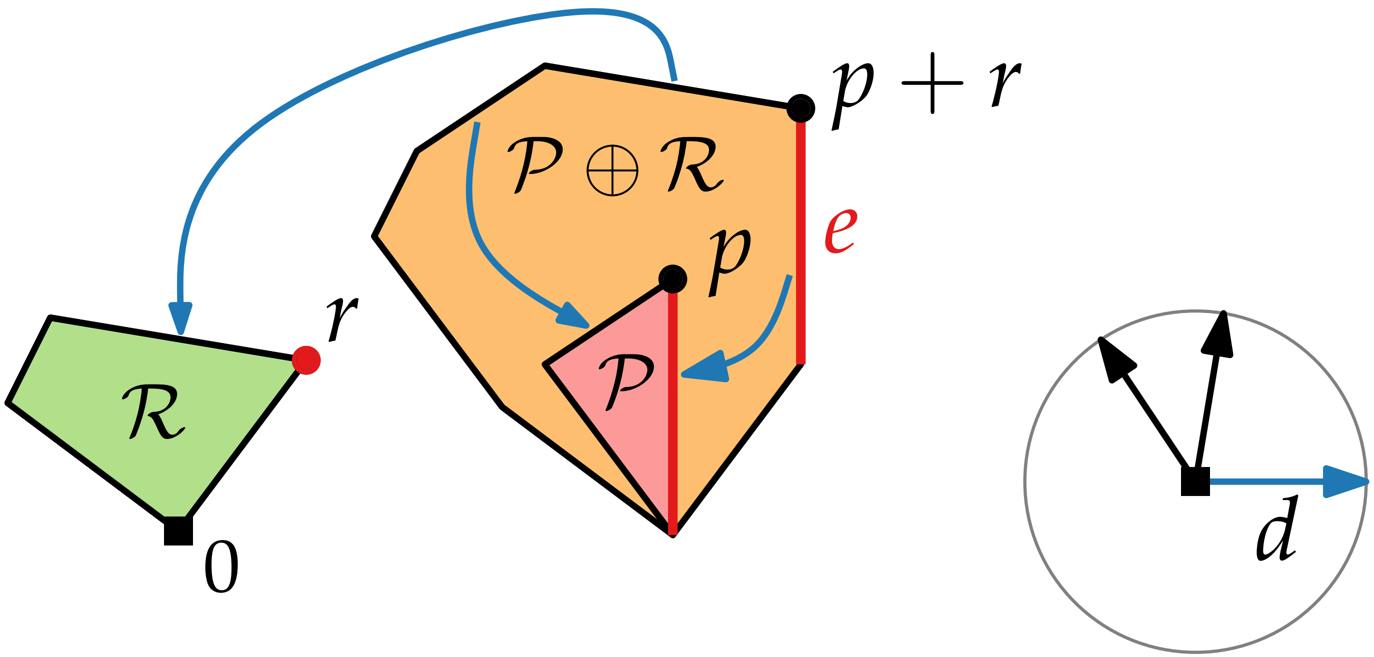
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



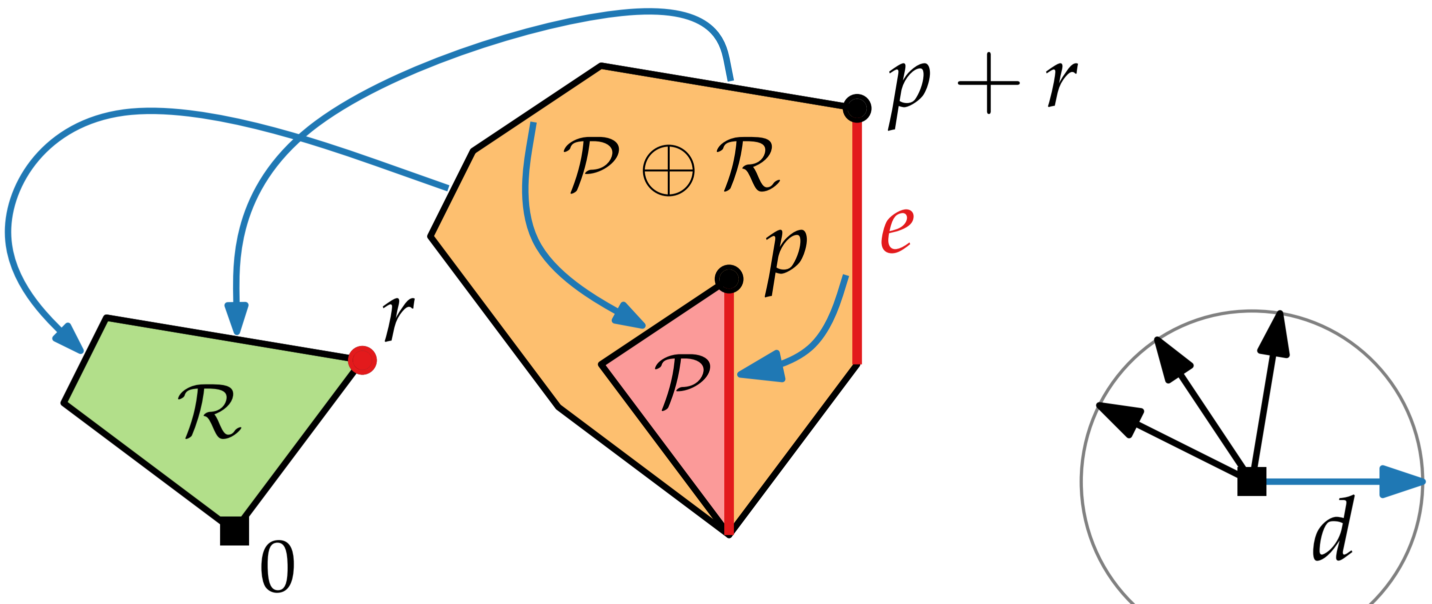
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



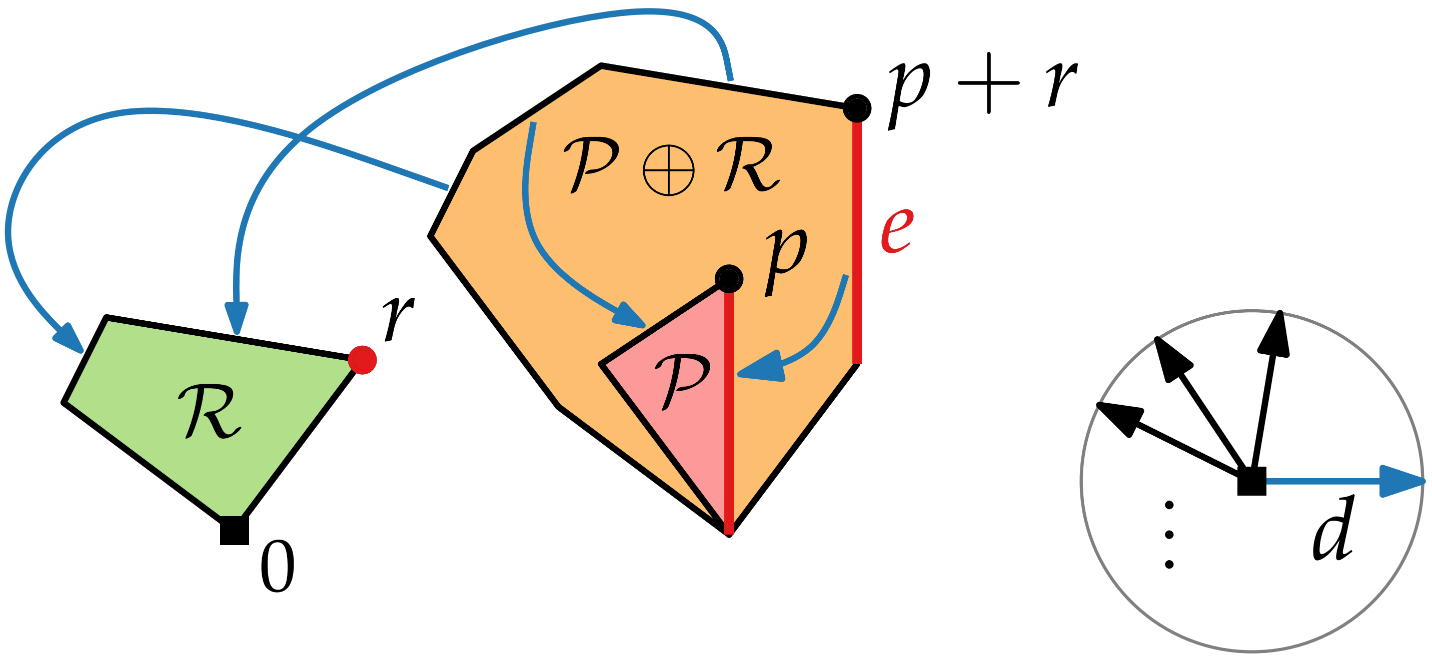
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.

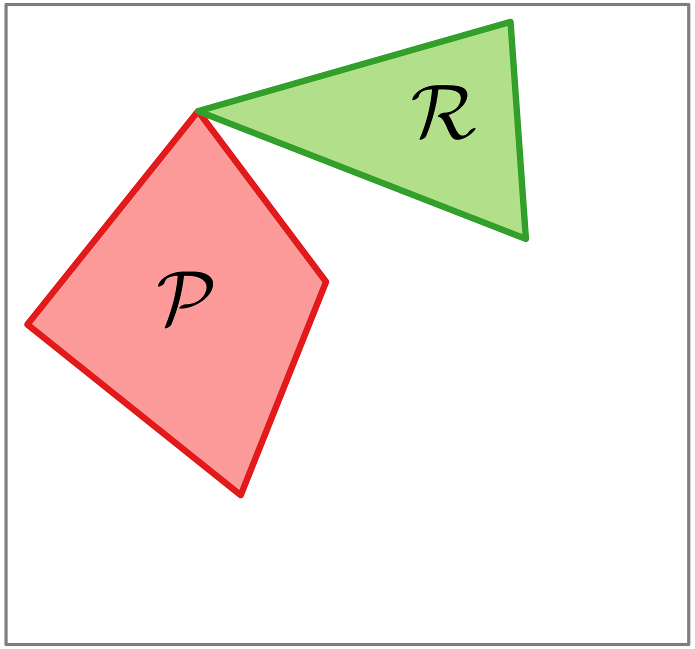


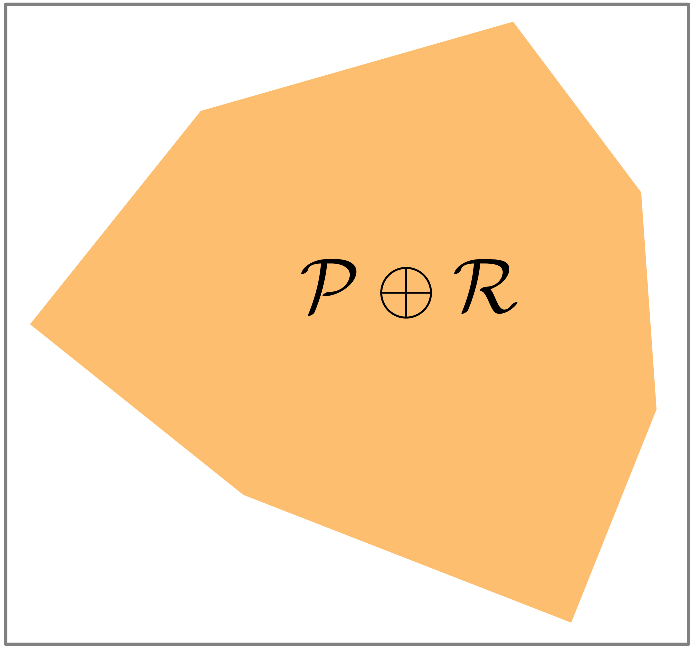
Minkowski Sums: Complexity

Theorem. If $\mathcal{P}$ and $\mathcal{R}$ are convex polygons with n and m edges, respectively, then $\mathcal{P} \oplus \mathcal{R}$ is a convex polygon with at most $n + m$ edges.



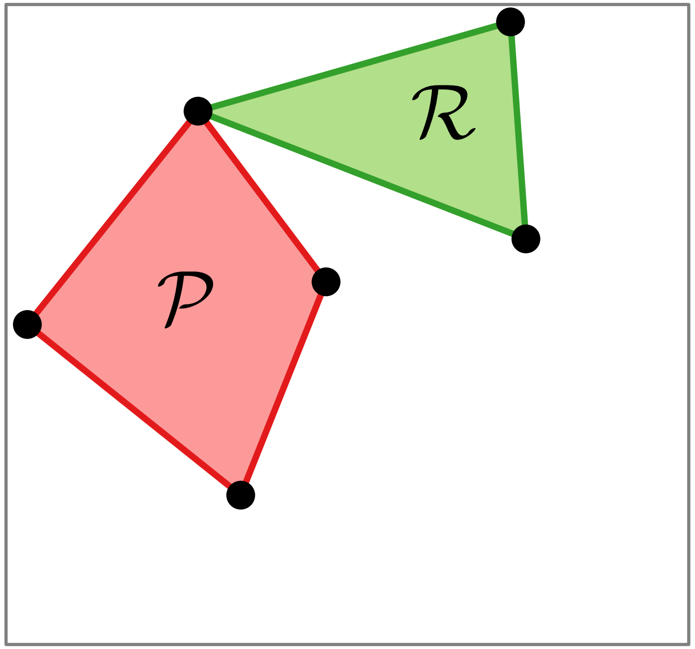
Minkowski Sums: Computation

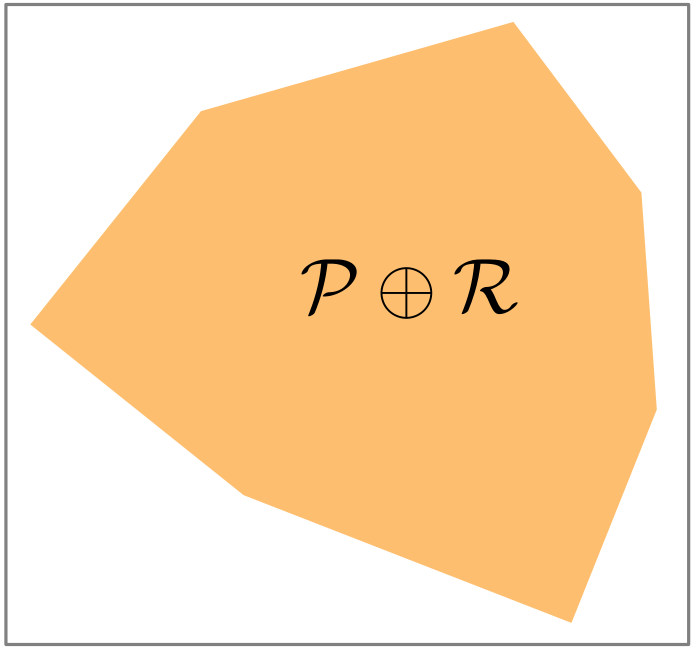




Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

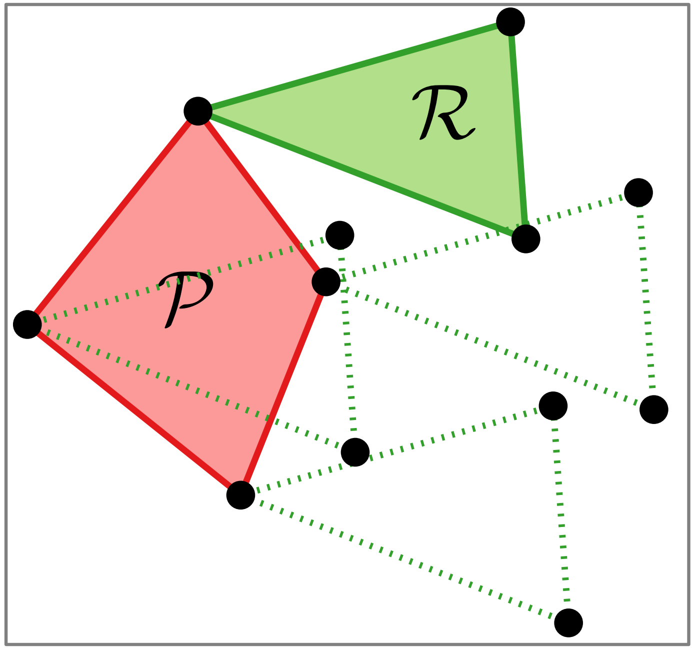


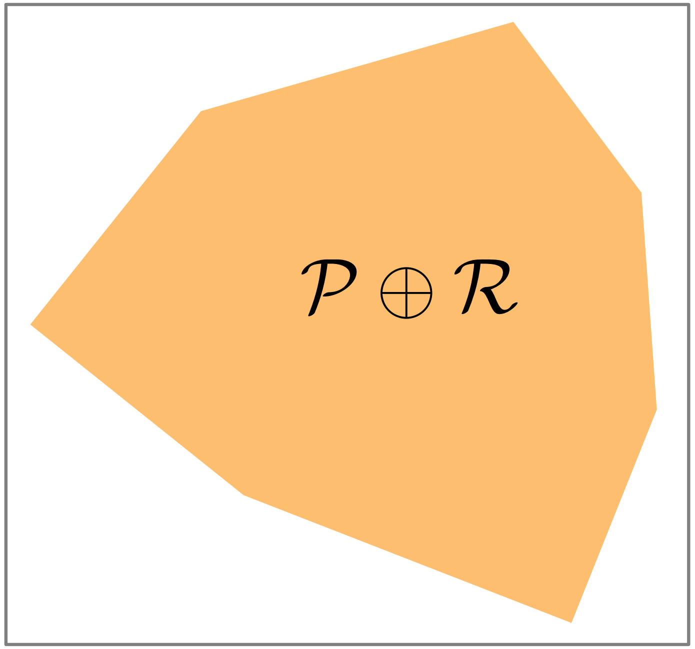


Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea.

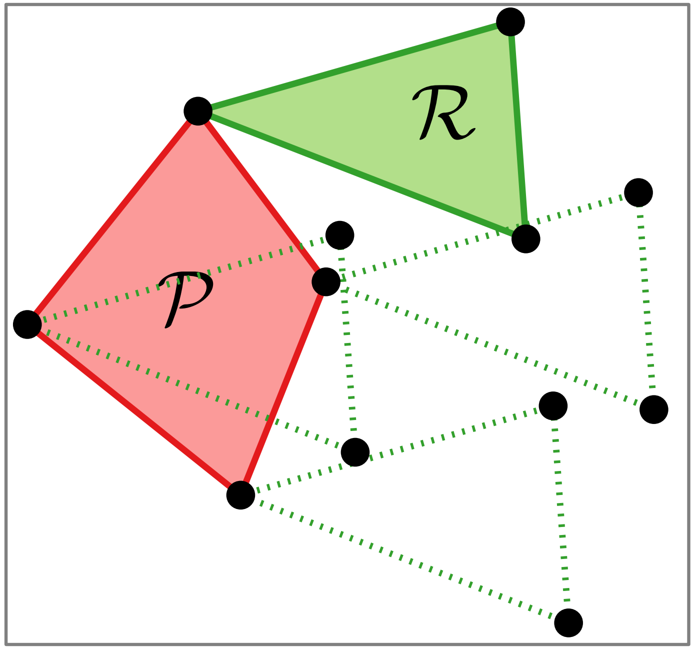


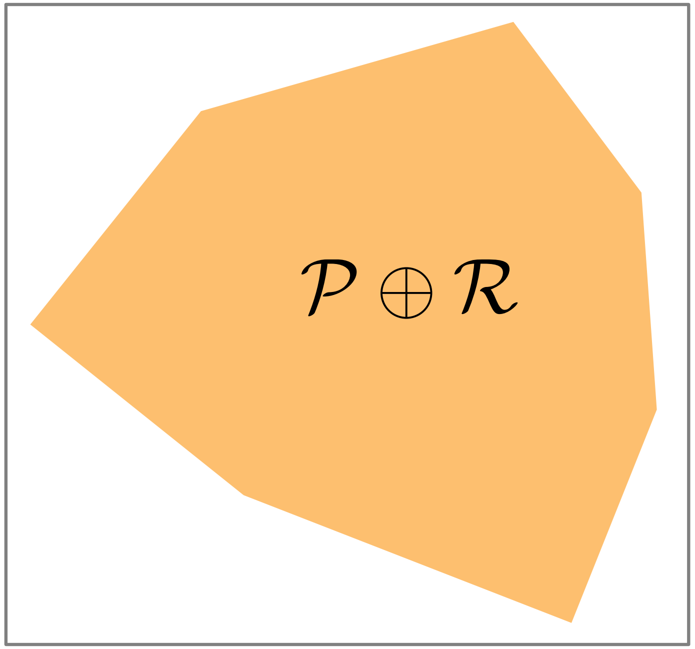


Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}\big)$ *(Proof?)*

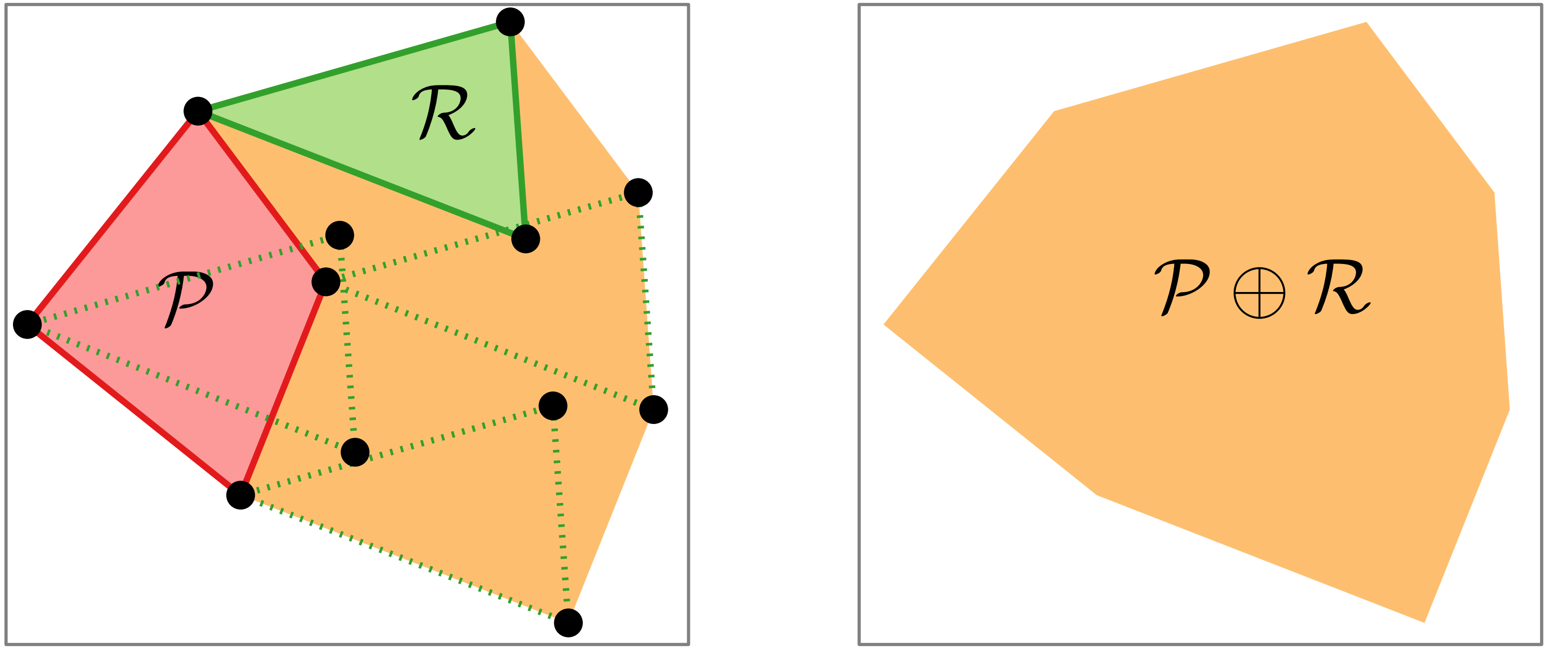




Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}\big)$ *(Proof?)*

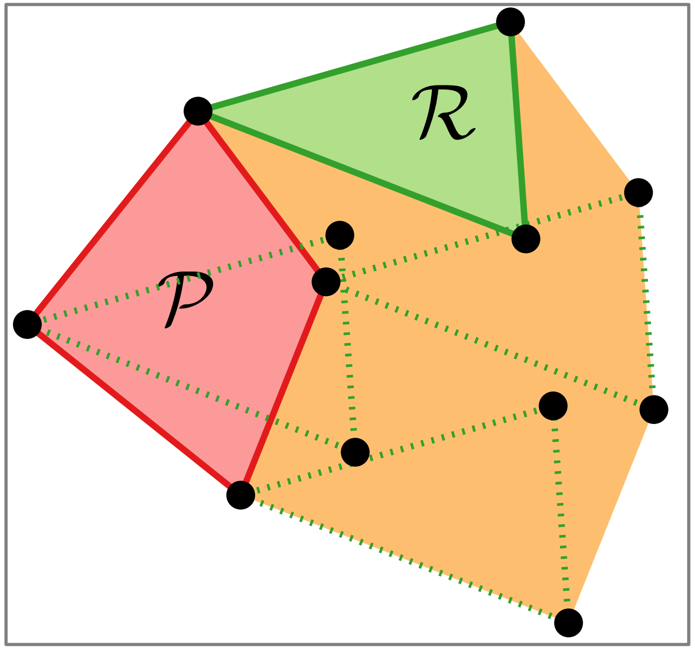


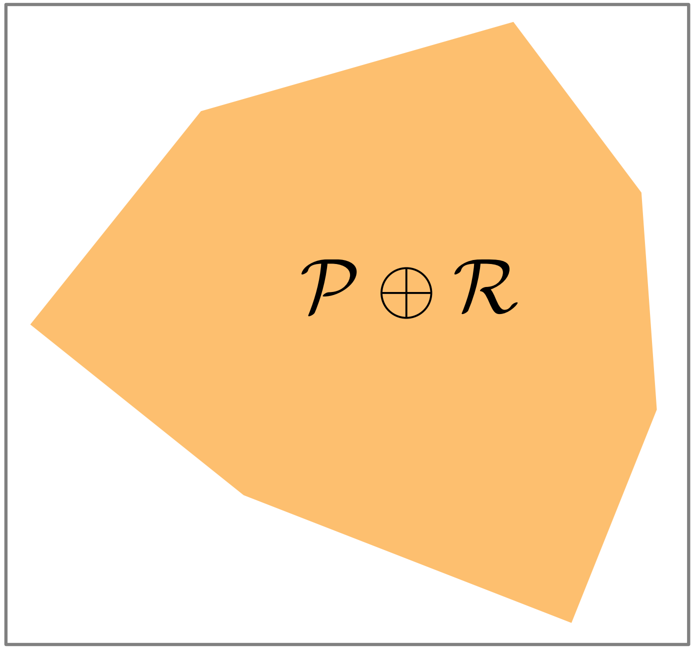
Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}\big)$ *(Proof?)*

Problem.



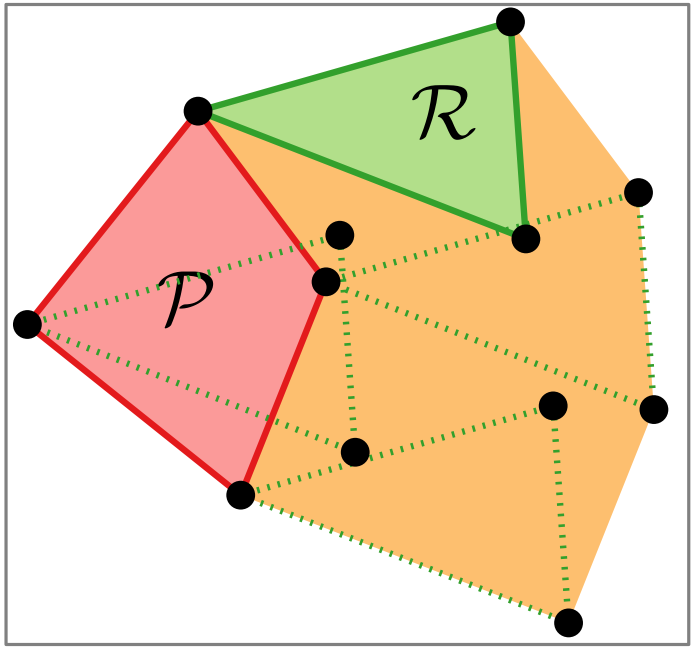


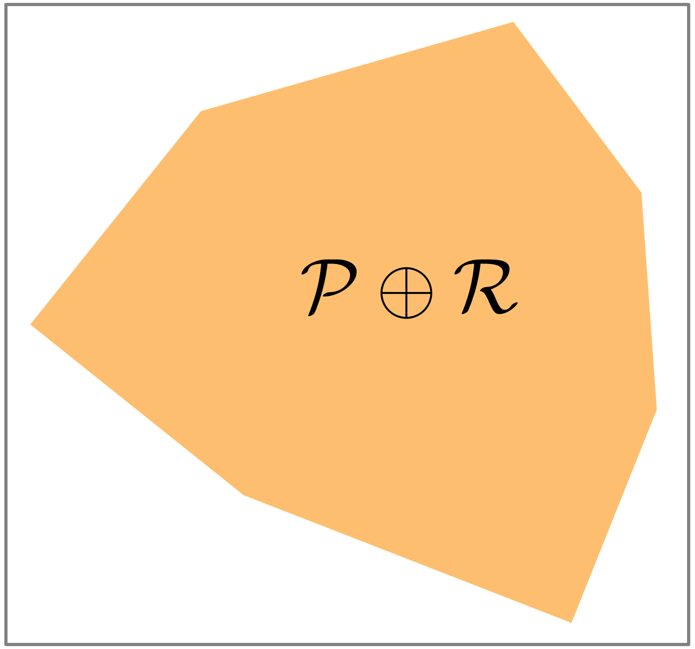
Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}_{\text{complexity} \in \Theta(\quad)}\big)$ *(Proof?)*

Problem.



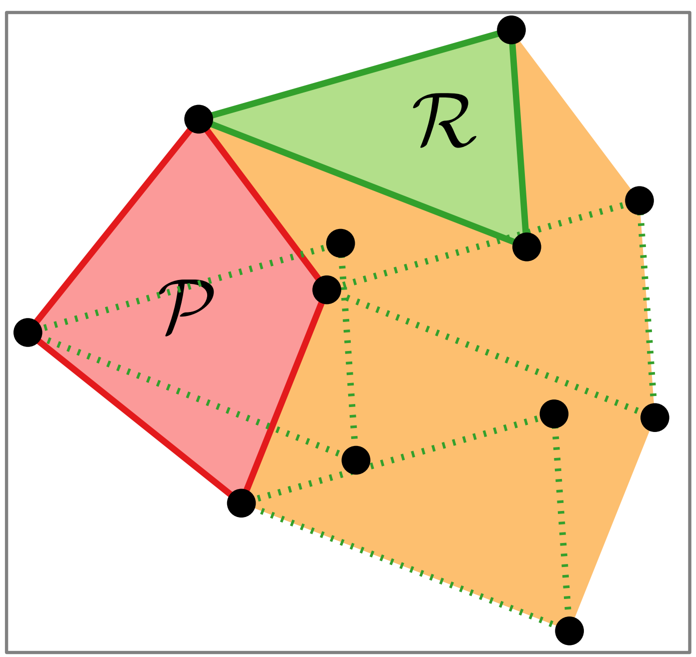


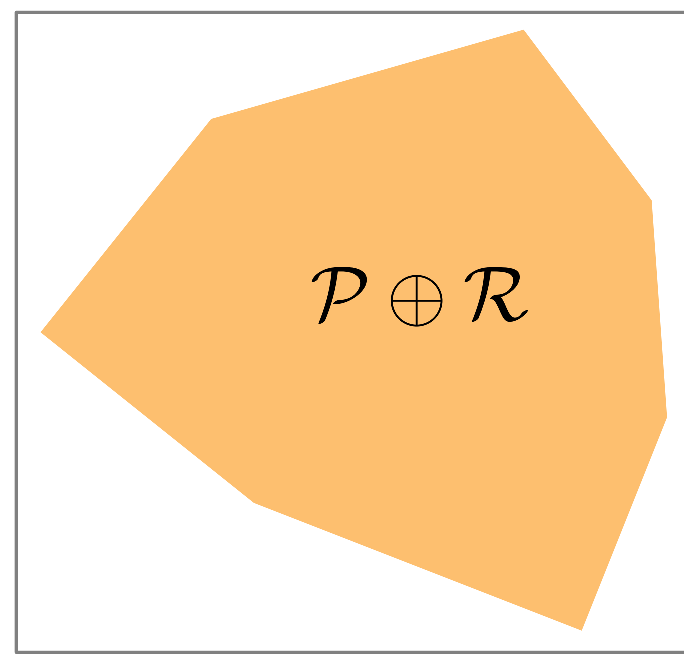
Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}_{\text{complexity} \in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(





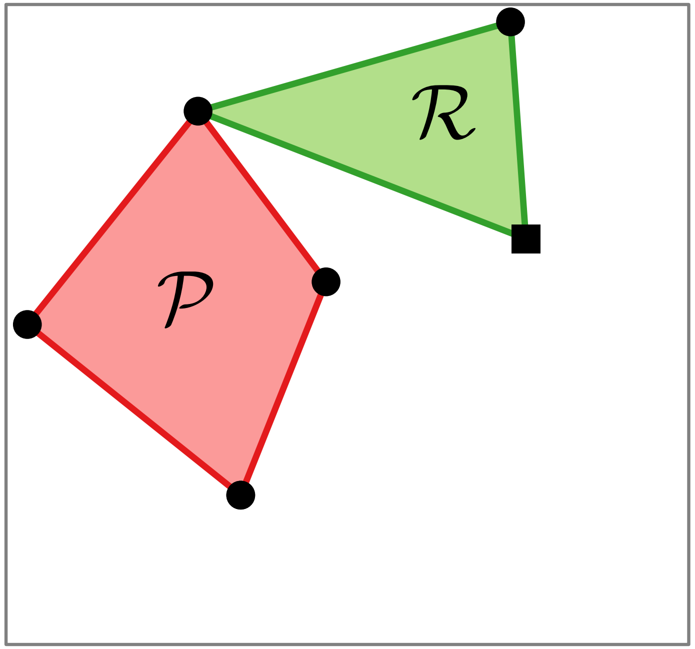
Minkowski Sums: Computation

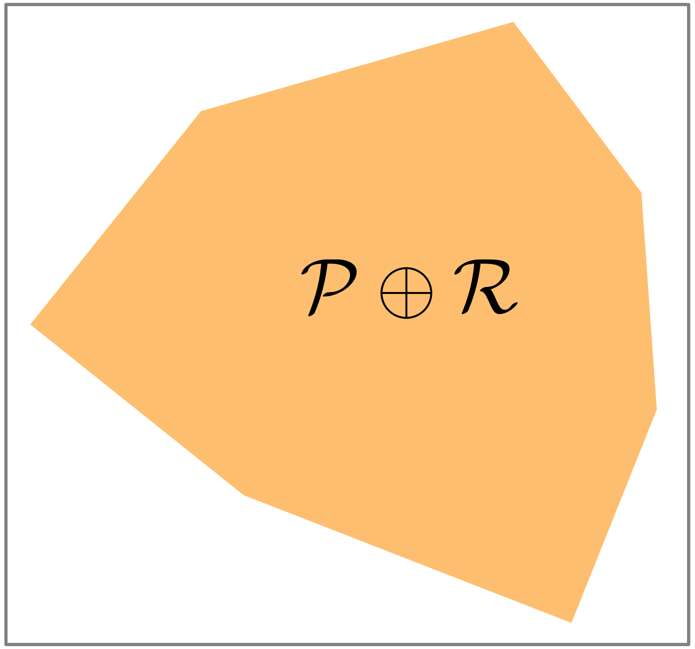
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





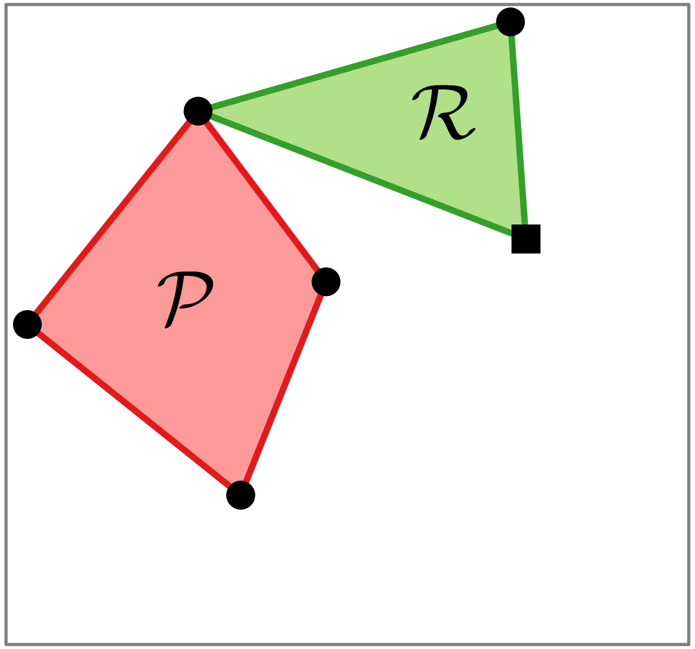
Minkowski Sums: Computation

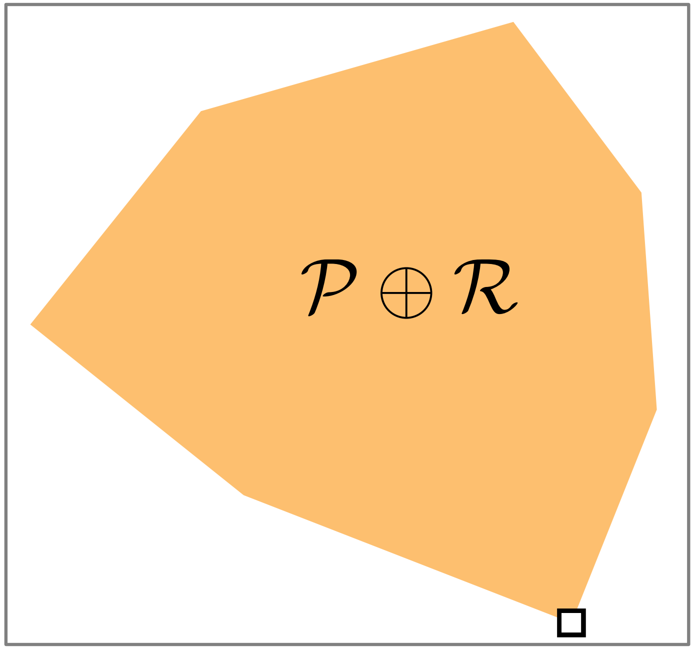
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





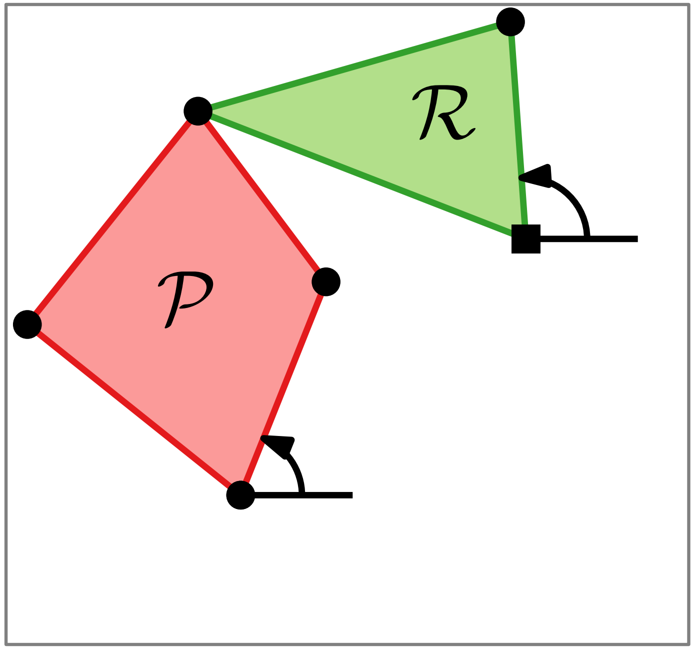
Minkowski Sums: Computation

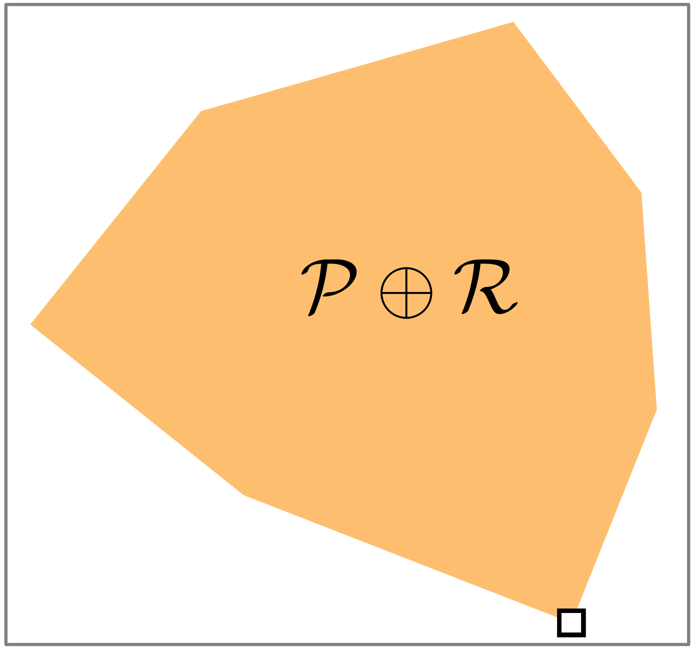
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$ **:-(**

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





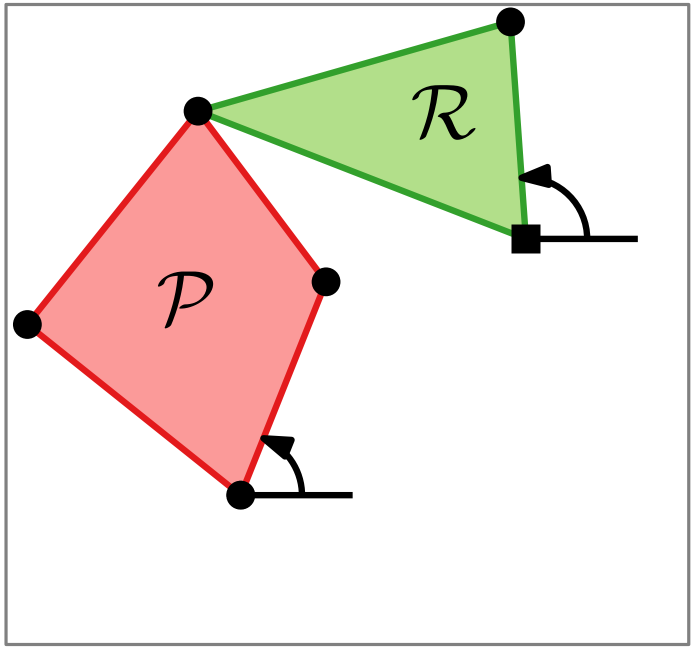
Minkowski Sums: Computation

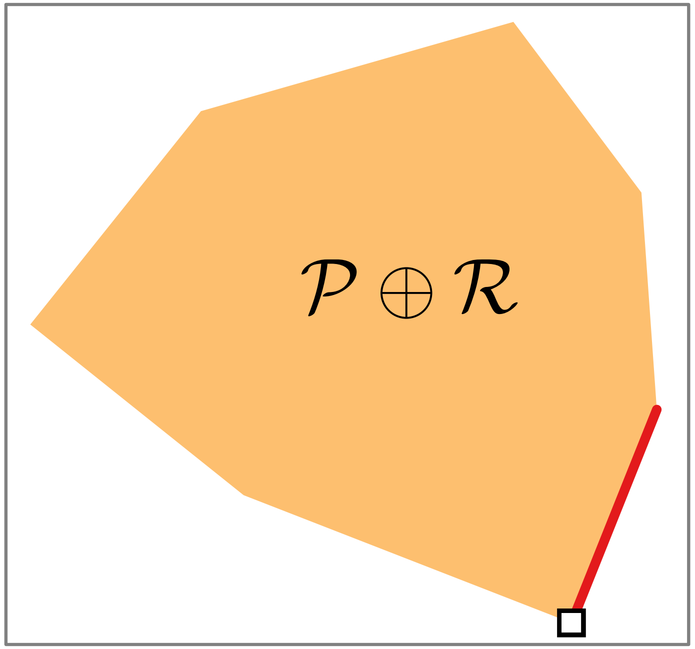
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$ **:-(**

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





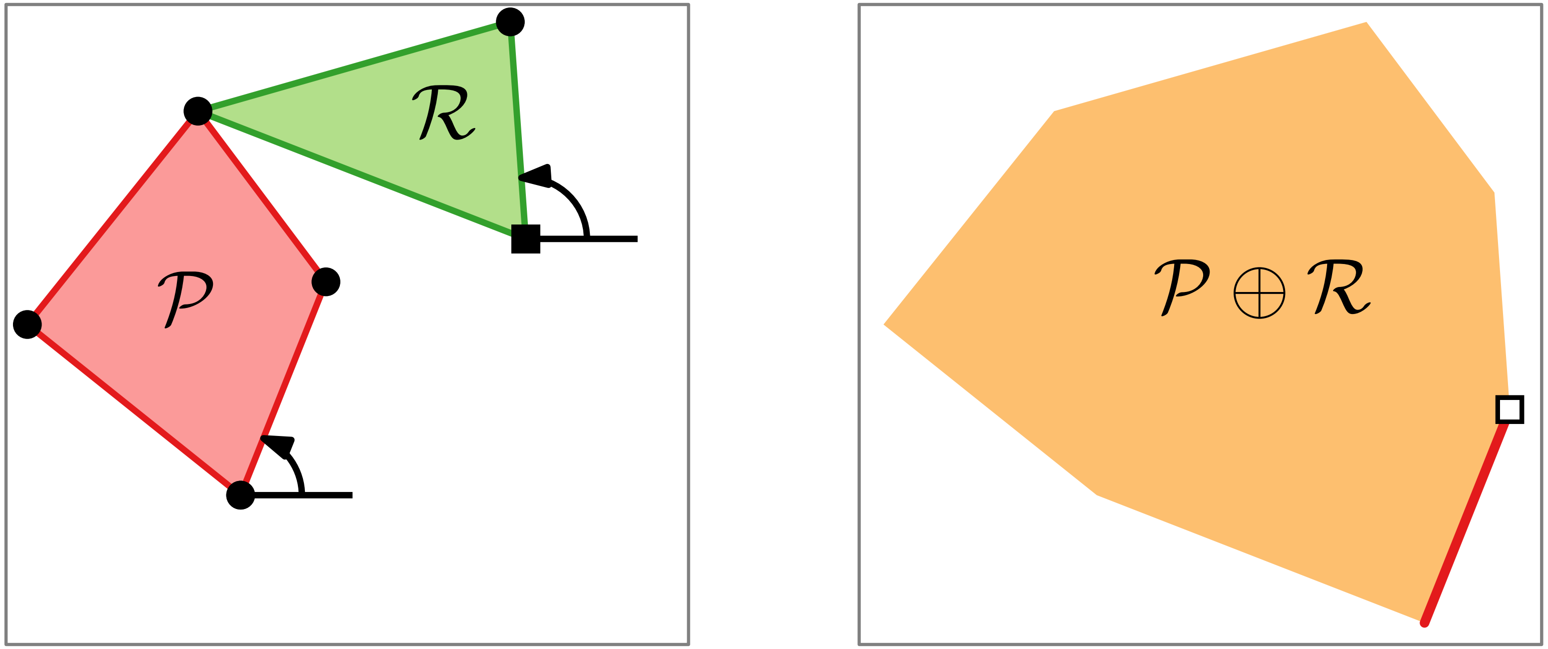
Minkowski Sums: Computation

Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.



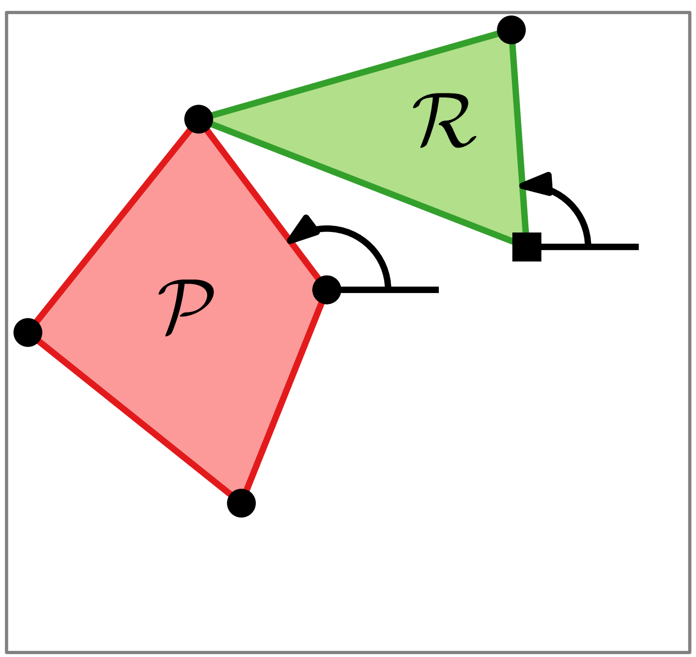
Minkowski Sums: Computation

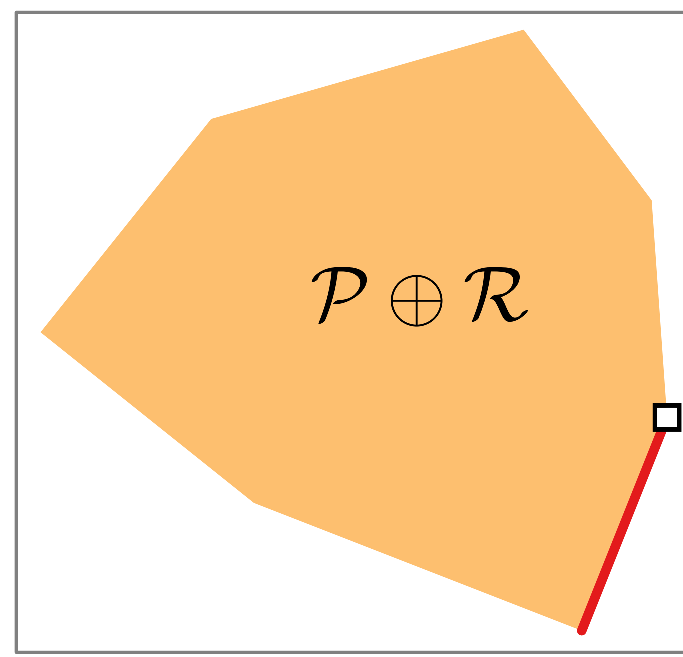
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





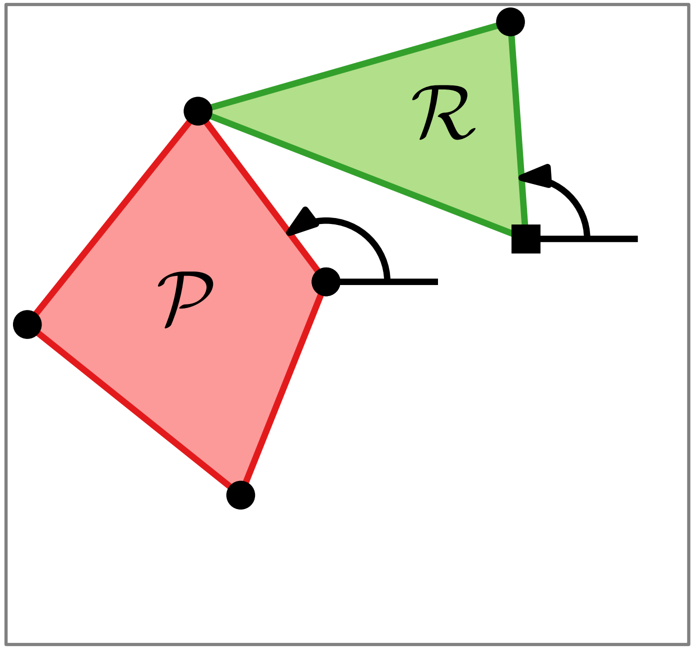
Minkowski Sums: Computation

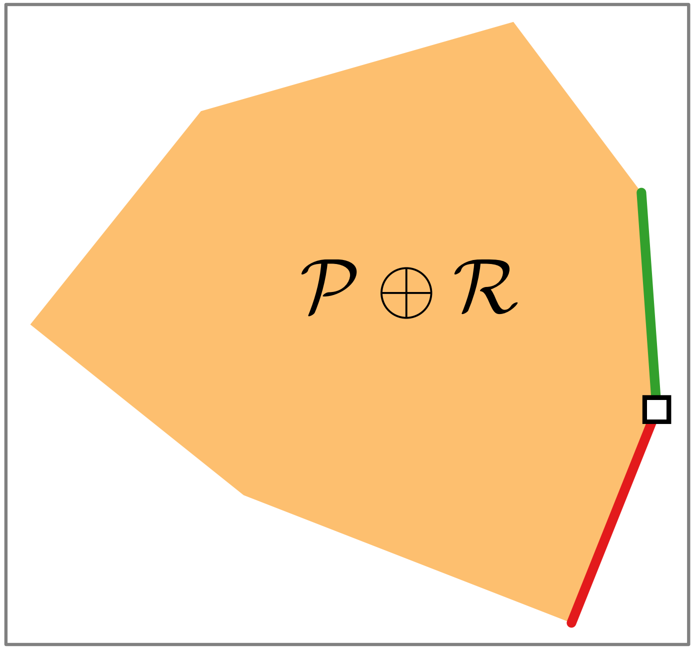
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





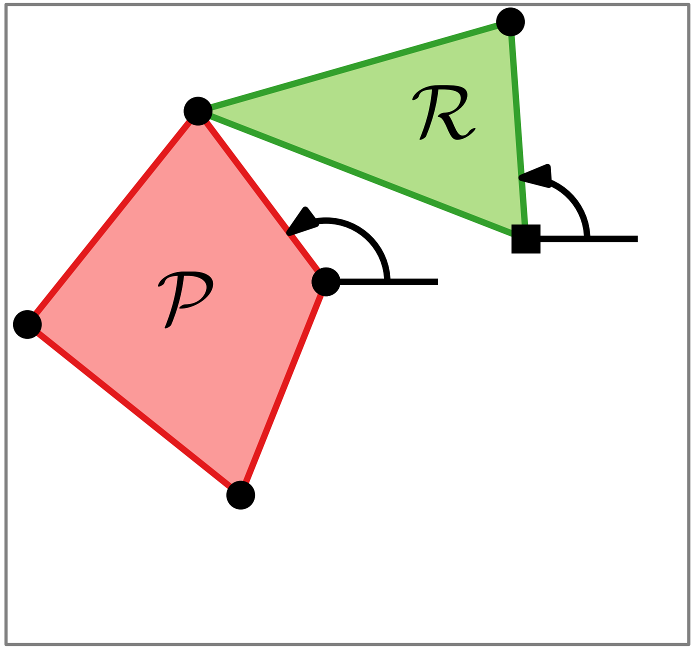
Minkowski Sums: Computation

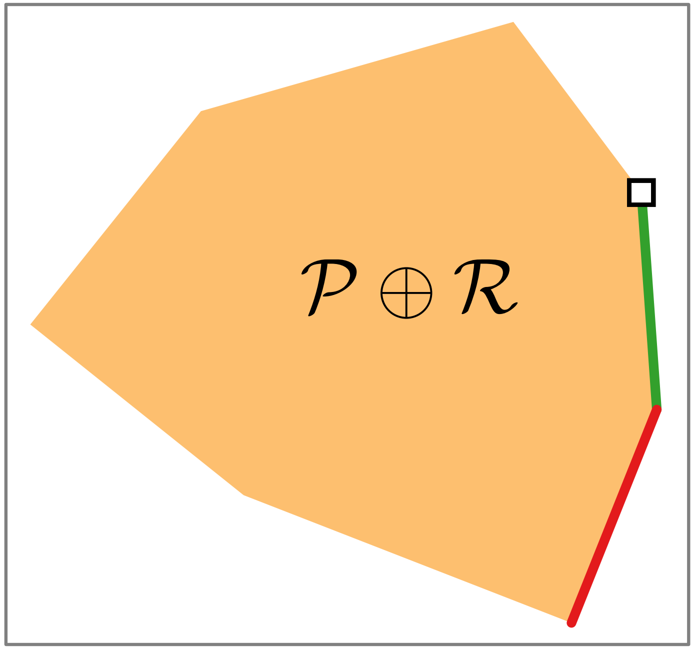
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\Big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\Big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





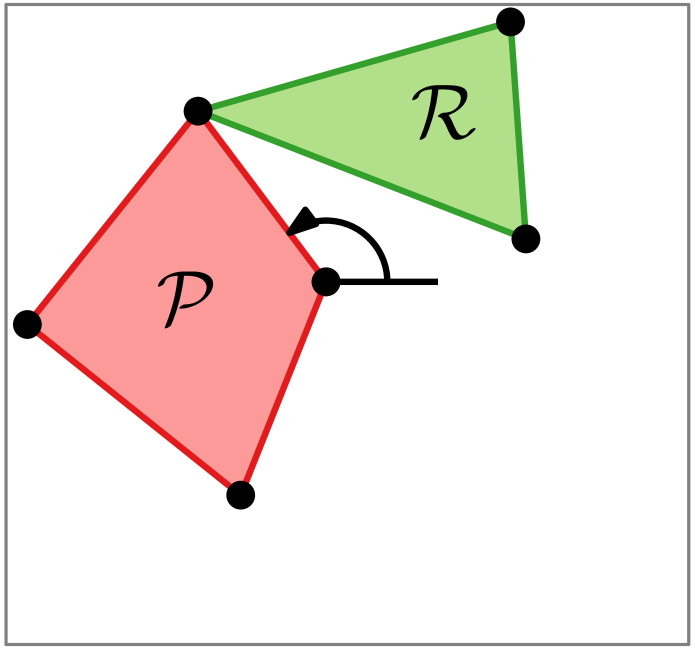
Minkowski Sums: Computation

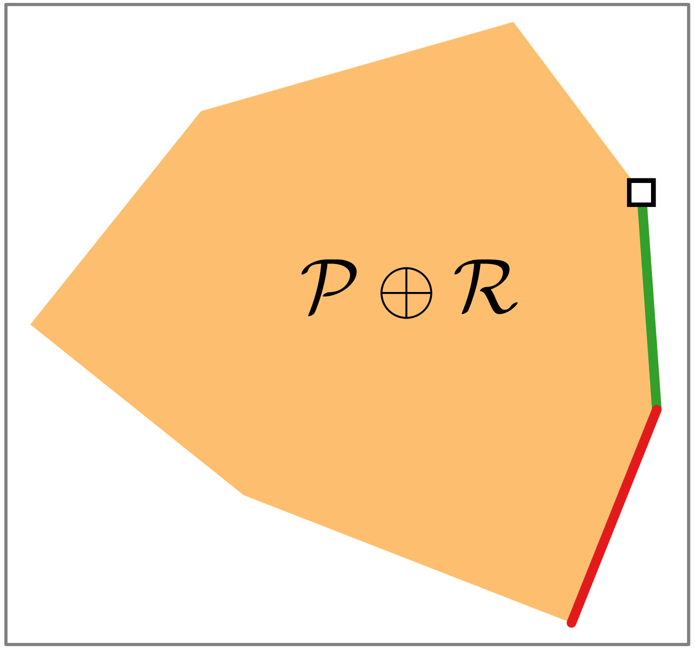
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





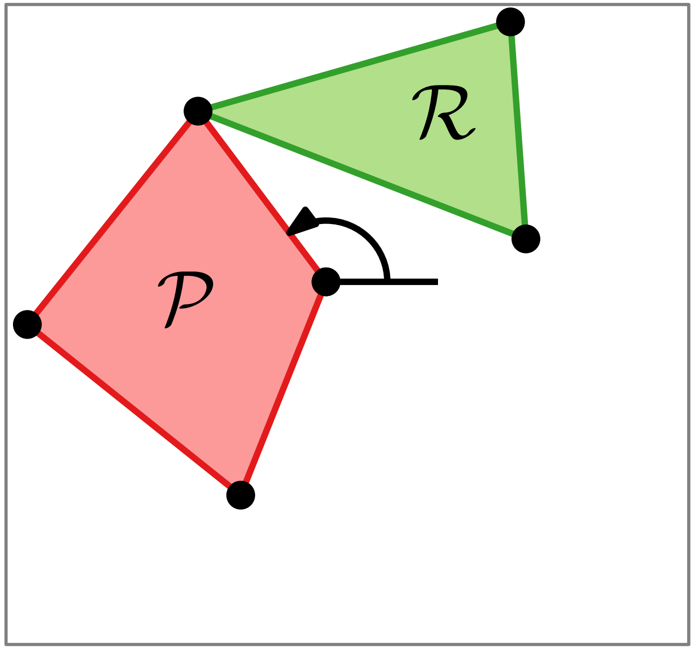
Minkowski Sums: Computation

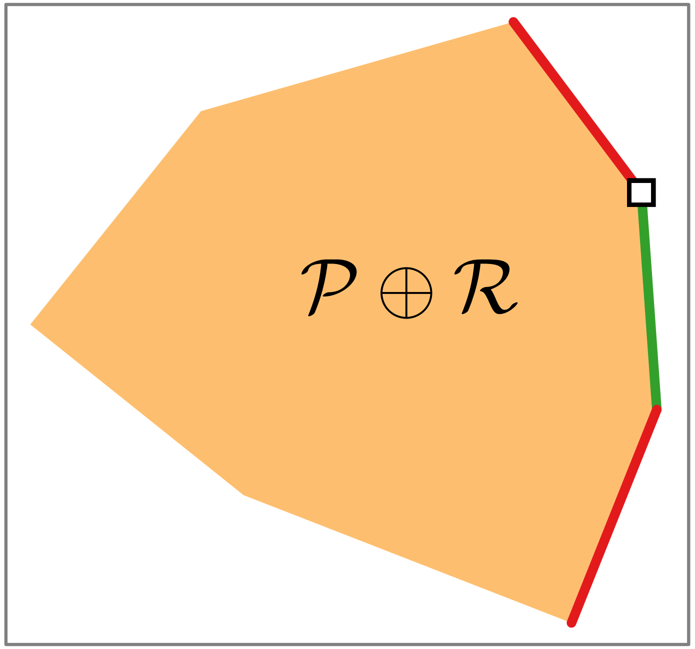
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





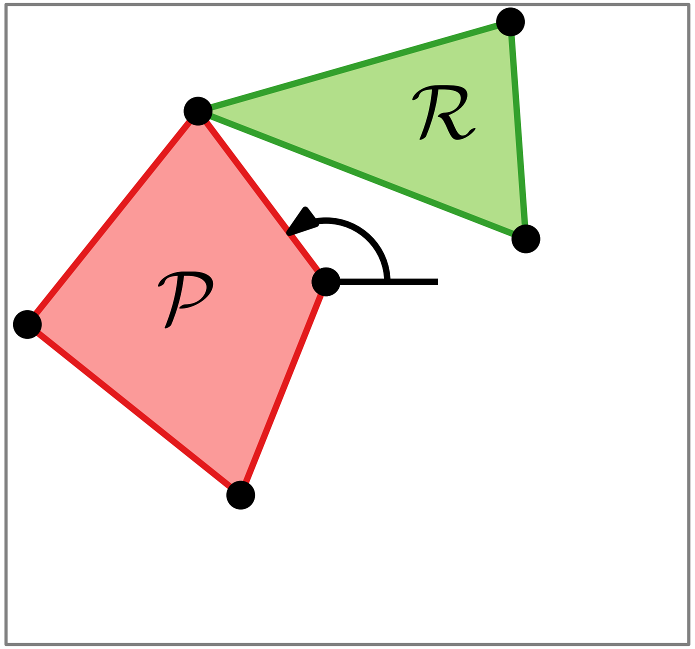
Minkowski Sums: Computation

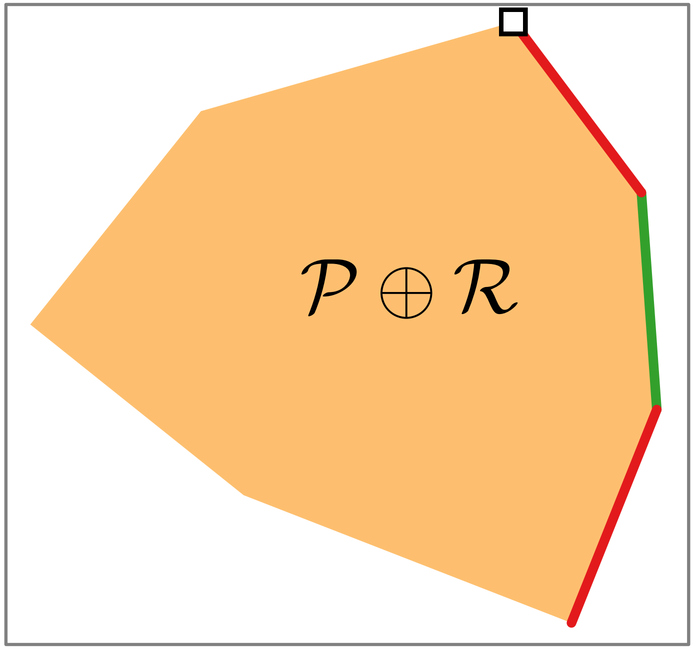
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





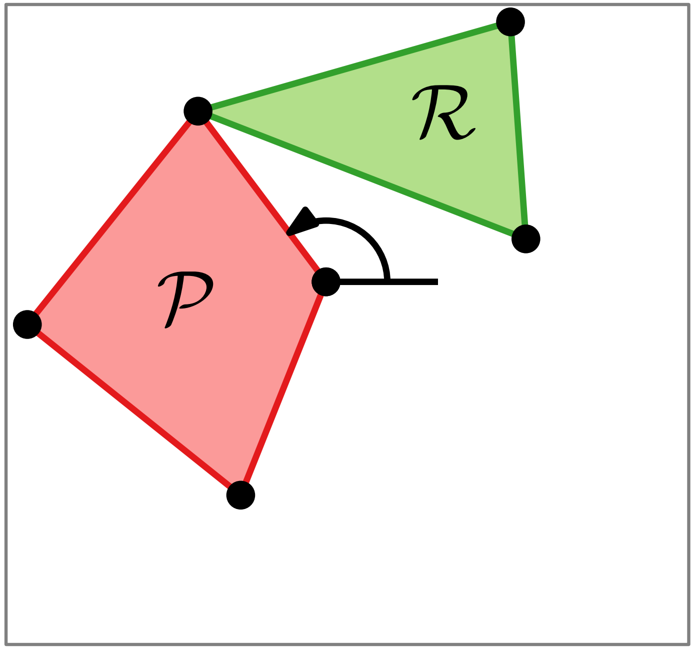
Minkowski Sums: Computation

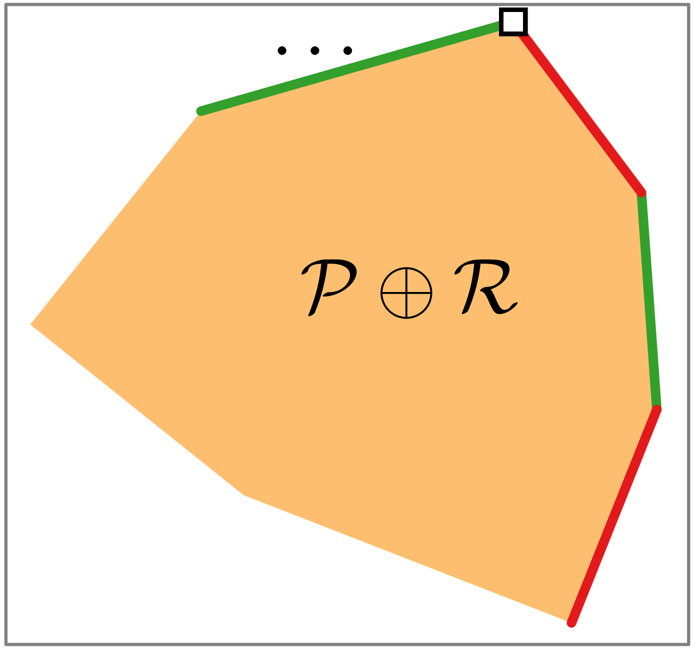
Task. How would you compute $\mathcal{P} \oplus \mathcal{R}$ given $\mathcal{P}$ and $\mathcal{R}$?

Idea. $\mathcal{P} \oplus \mathcal{R} = \mathrm{CH}\big(\underbrace{\{p + r \mid p \in \mathcal{P}, r \in \mathcal{R}\}}\big)$ *(Proof?)*

Problem. complexity $\in \Theta(|\mathcal{P}| \cdot |\mathcal{R}|)$:-(

Theorem. The Minkowski sum of two convex polygons $\mathcal{P}$ and $\mathcal{R}$ can be computed in $O(|\mathcal{P}| + |\mathcal{R}|)$ time.





Computational Geometry

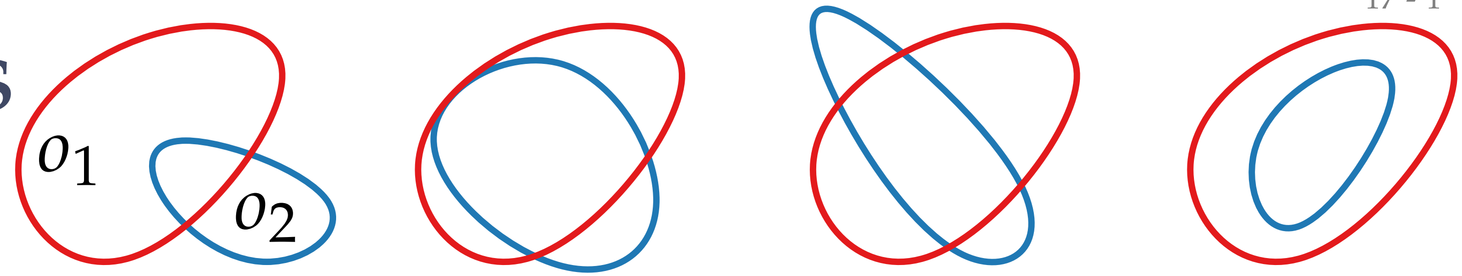
Lecture 10:
Motion Planning

Part V:
Pseudodisks

Philipp Kindermann

Winter Semester 2020

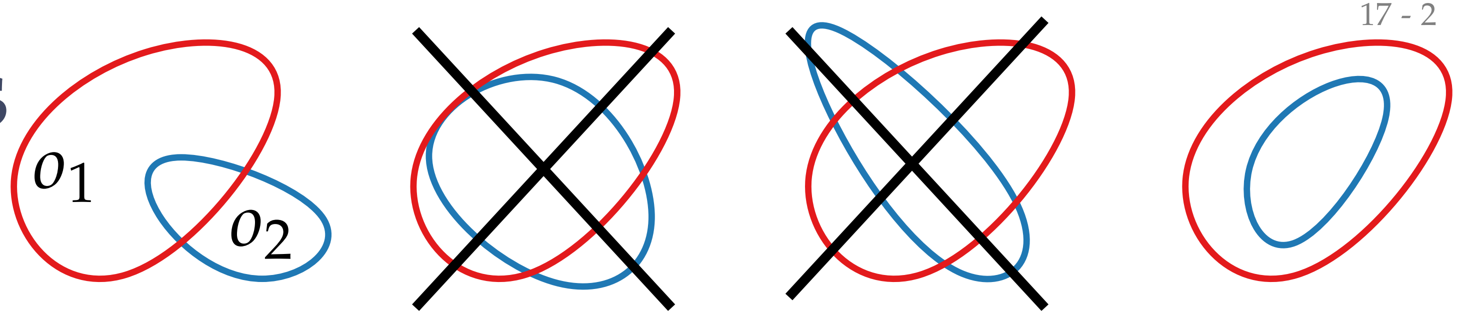
Pseudodisks



Definition. A pair of planar objects o_1 and o_2 is a pair of pseudodisks if:

- $\partial o_1 \cap \operatorname{int}(o_2)$ is connected, and
- $\partial o_2 \cap \operatorname{int}(o_1)$ is connected.

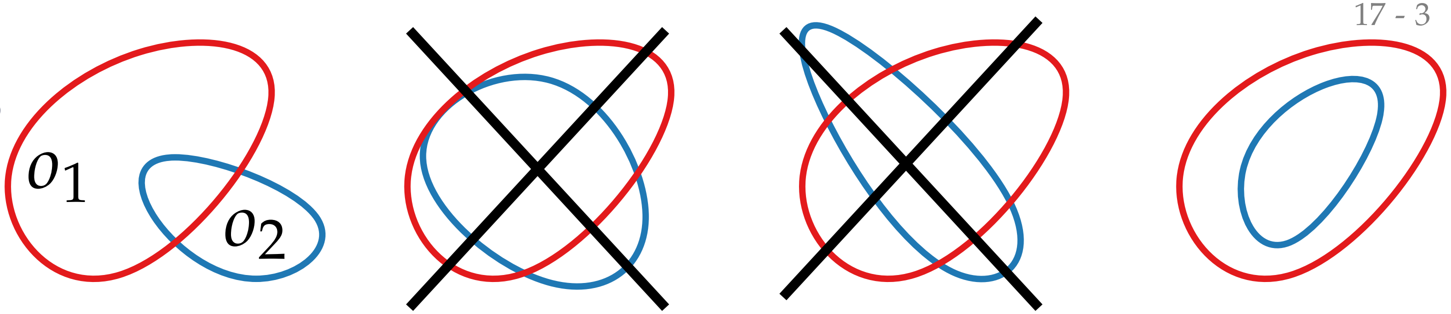
Pseudodisks



Definition. A pair of planar objects o_1 and o_2 is a pair of pseudodisks if:

- $\partial o_1 \cap \text{int}(o_2)$ is connected, and
- $\partial o_2 \cap \text{int}(o_1)$ is connected.

Pseudodisks

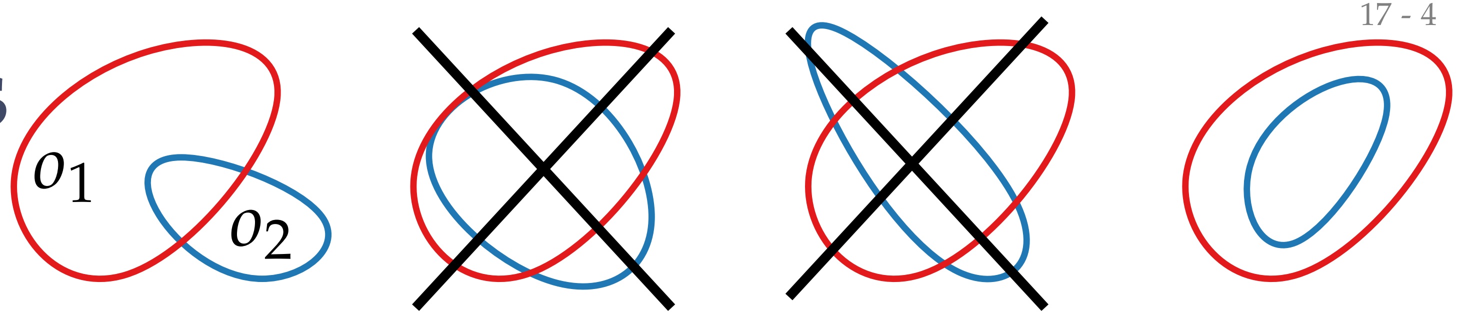


Definition. A pair of planar objects o_1 and o_2 is a pair of pseudodisks if:

- $\partial o_1 \cap \text{int}(o_2)$ is connected, and
- $\partial o_2 \cap \text{int}(o_1)$ is connected.

$p \in \partial o_1 \cap \partial o_2$ is a *boundary crossing* if ∂o_1 crosses at p from the interior to the exterior of o_2.

Pseudodisks



Definition. A pair of planar objects o_1 and o_2 is a pair of pseudodisks if:

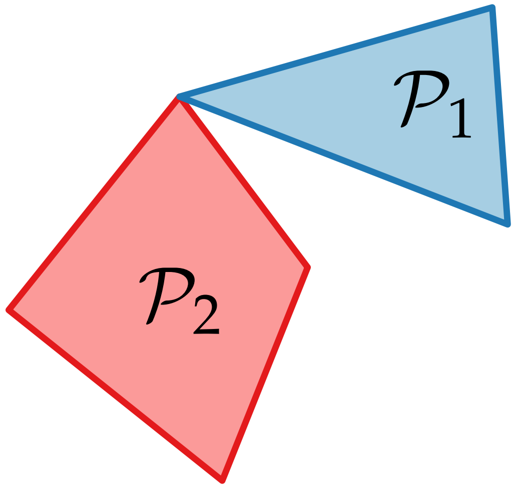
- $\partial o_1 \cap \operatorname{int}(o_2)$ is connected, and
- $\partial o_2 \cap \operatorname{int}(o_1)$ is connected.

$p \in \partial o_1 \cap \partial o_2$ is a *boundary crossing* if ∂o_1 crosses at p from the interior to the exterior of o_2.

Observation. A pair of polygonal pseudodisks defines at most two boundary crossings.

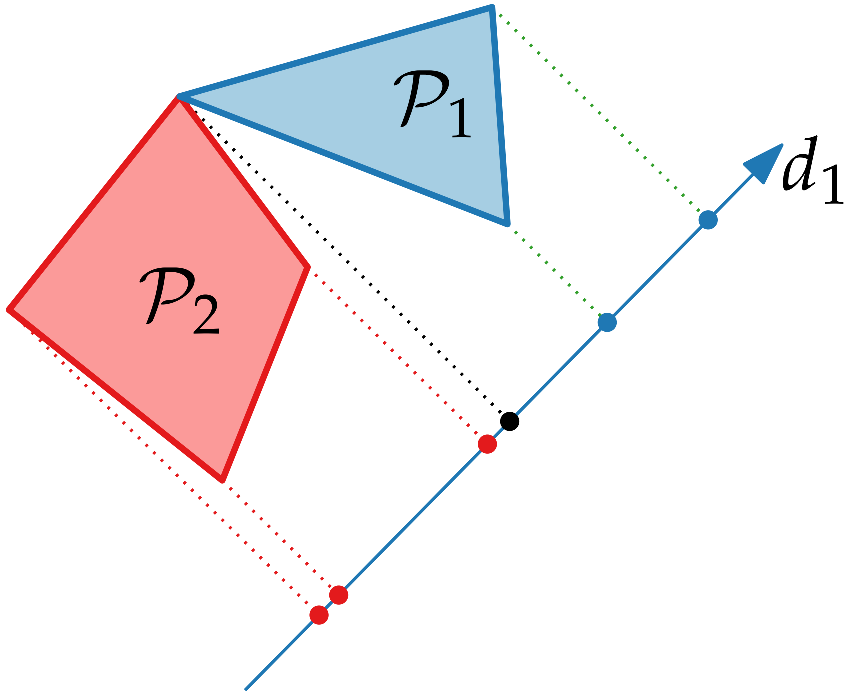
Extreme Directions

Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons



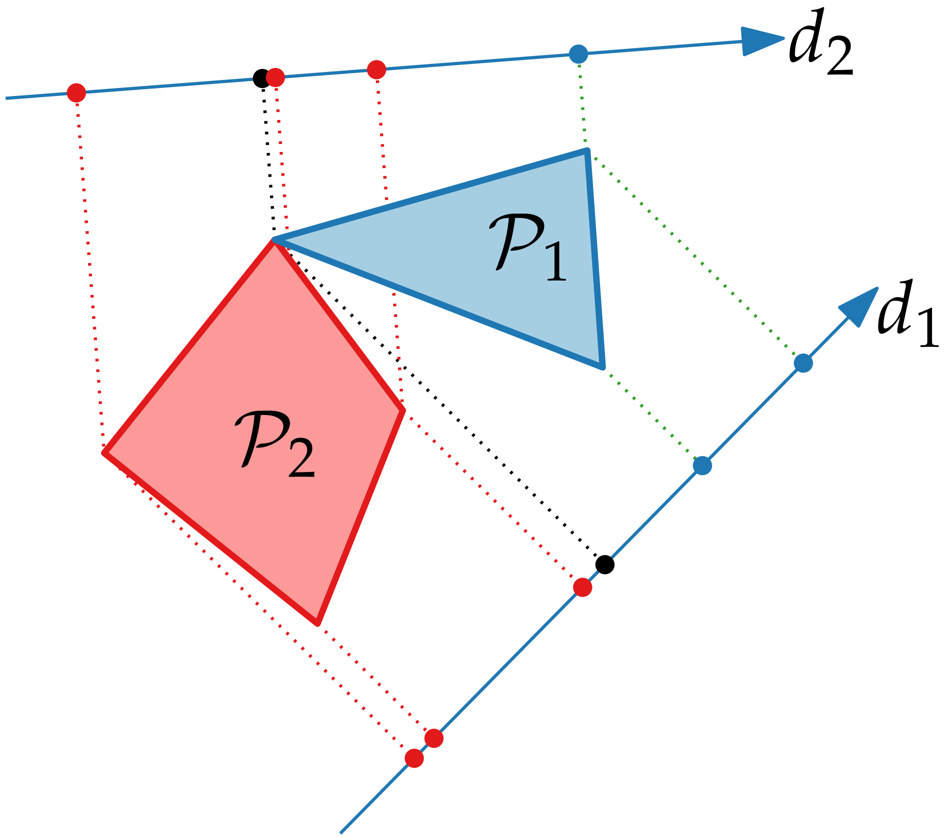
Extreme Directions

Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons. Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.



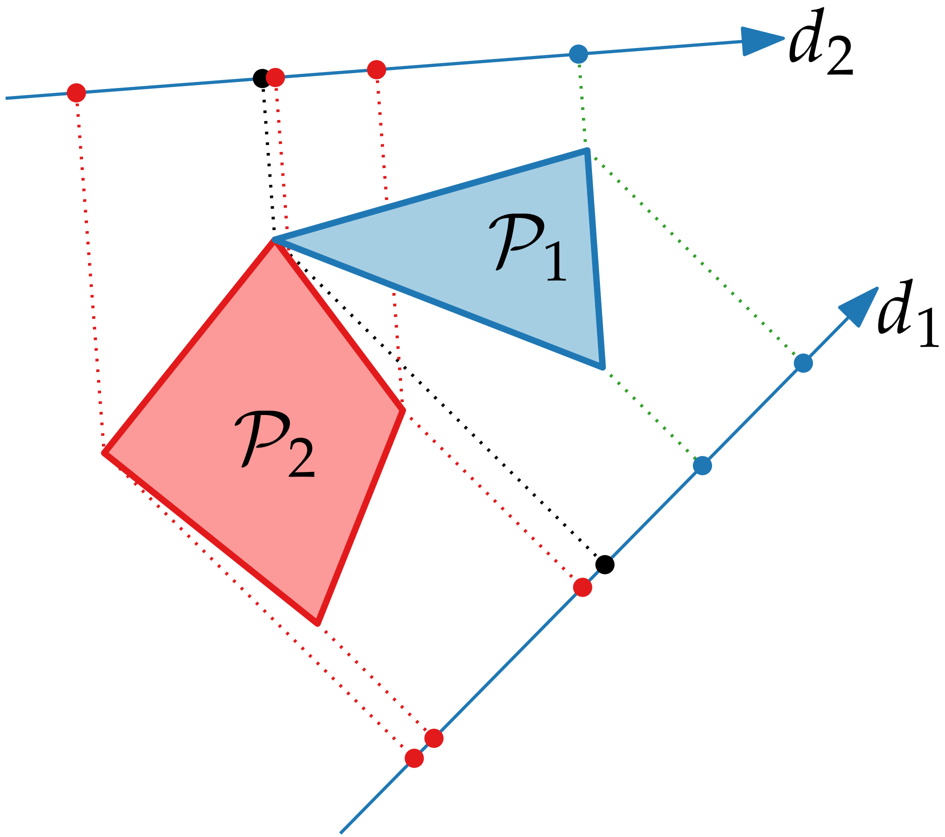
Extreme Directions

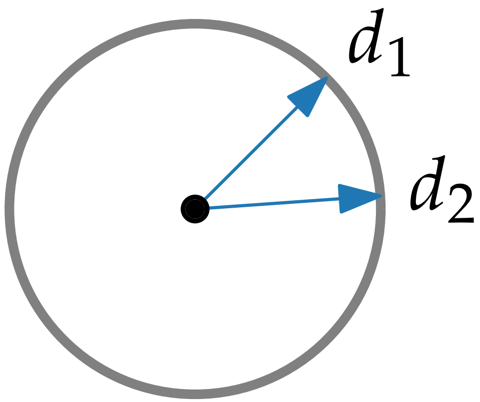
Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons. Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.



Extreme Directions

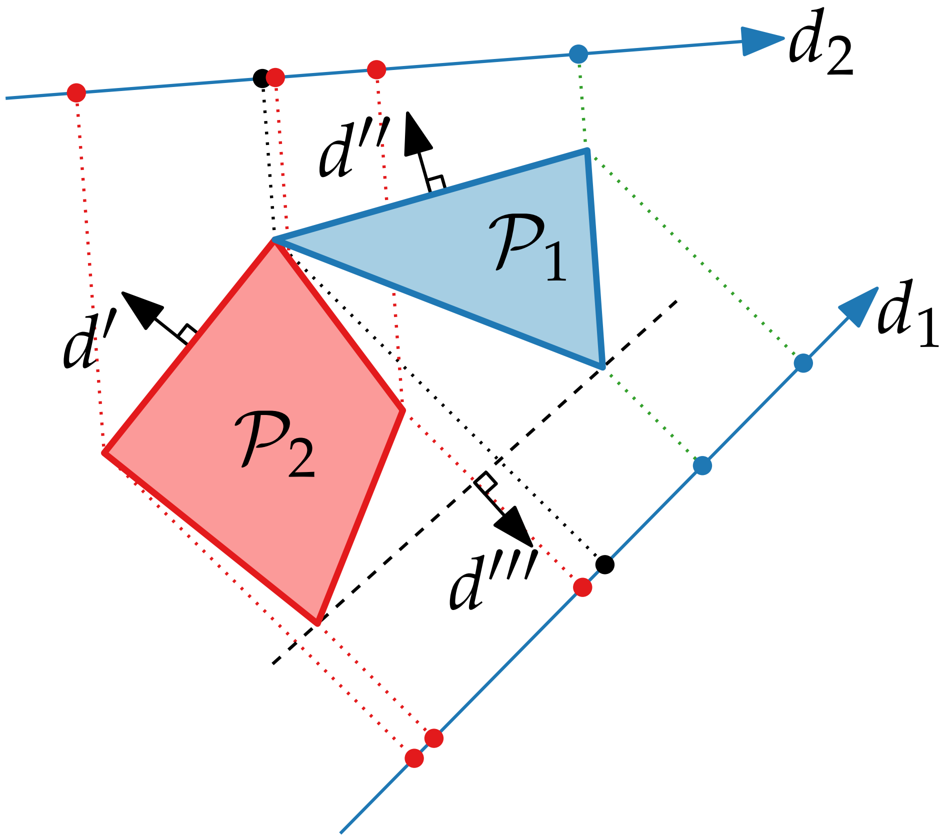
Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons. Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.
Then $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$ either in $[d_1, d_2]$ or in $[d_2, d_1]$.

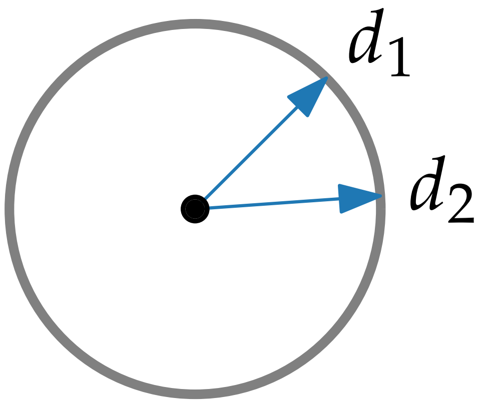




Extreme Directions

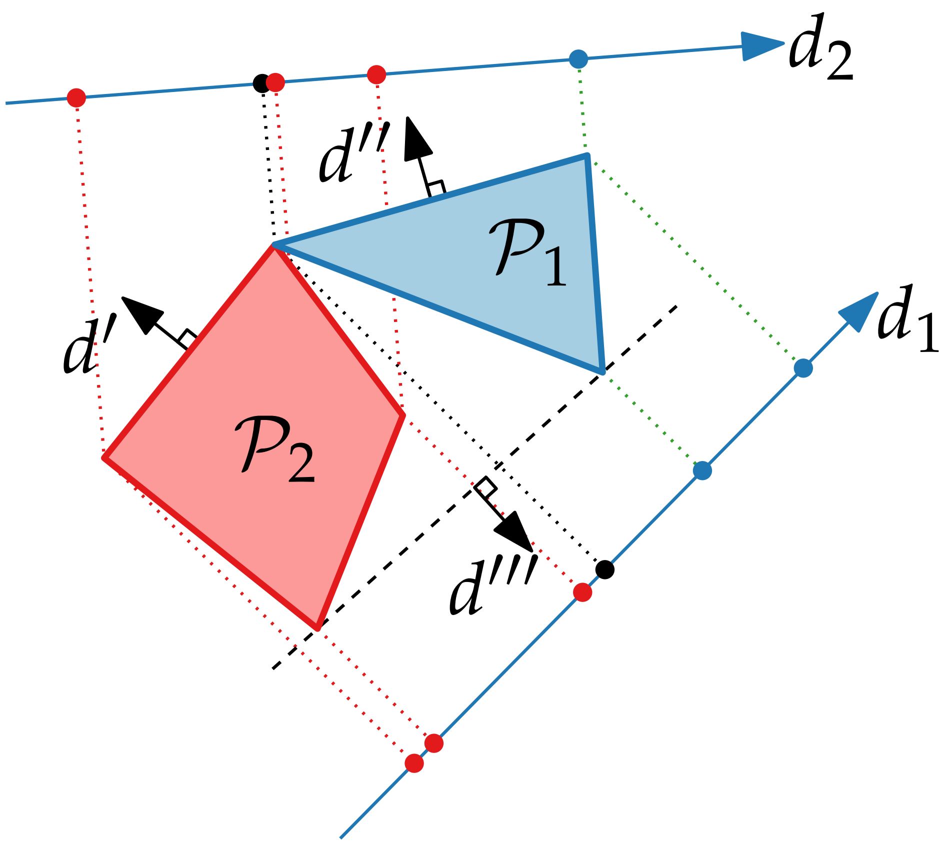
Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons
Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.
Then $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$ either in $[d_1, d_2]$ or in $[d_2, d_1]$.

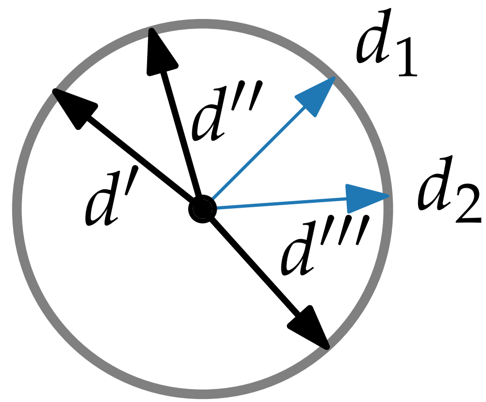




Extreme Directions

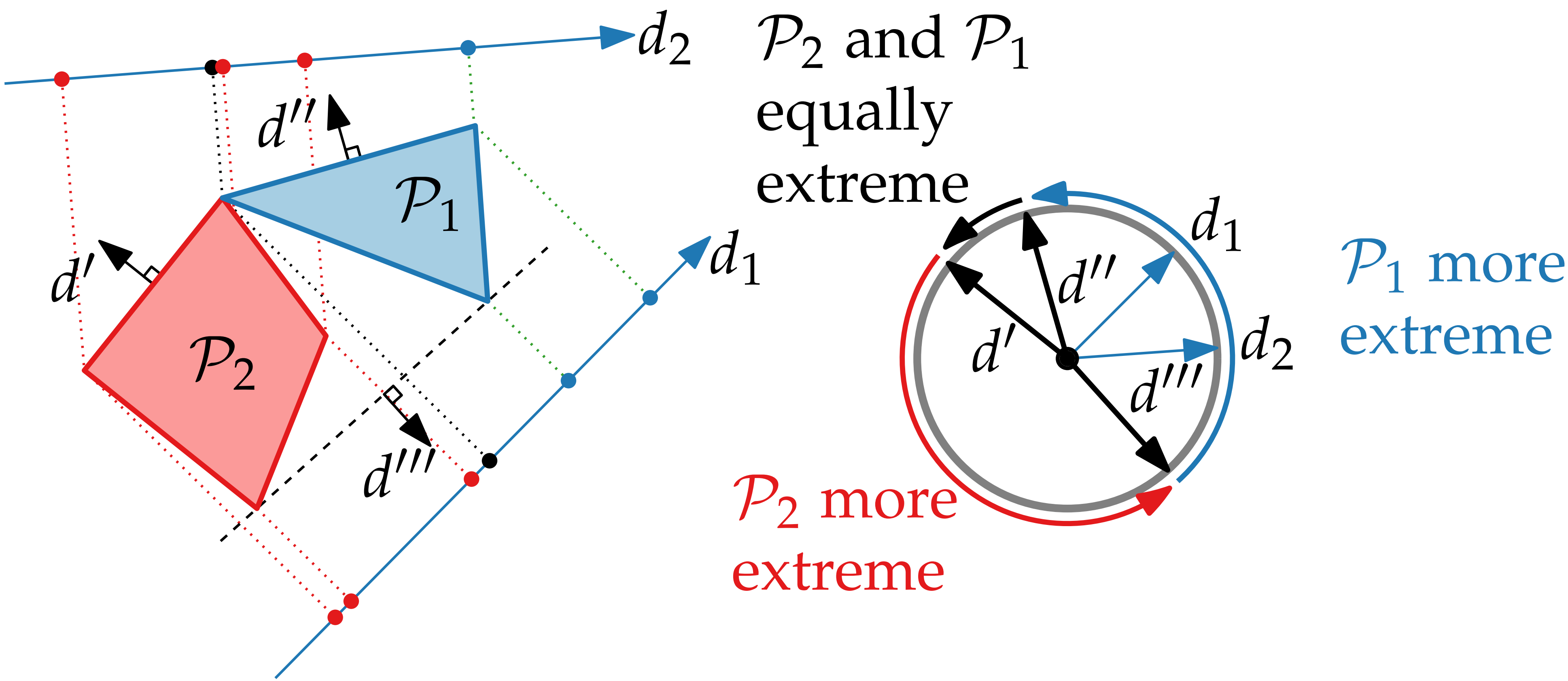
Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons. Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.
Then $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$ either in $[d_1, d_2]$ or in $[d_2, d_1]$.





Extreme Directions

Observation. Let $\mathcal{P}_1$, $\mathcal{P}_2$ be interior-disjoint convex polygons
Let d_1 and d_2 be directions in which $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$.
Then $\mathcal{P}_1$ is more extreme than $\mathcal{P}_2$ either in $[d_1, d_2]$ or in $[d_2, d_1]$.



Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\mathcal{P}_1 \oplus \mathcal{R}$ and $\mathcal{P}_2 \oplus \mathcal{R}$ is a pair of pseudodisks.

Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

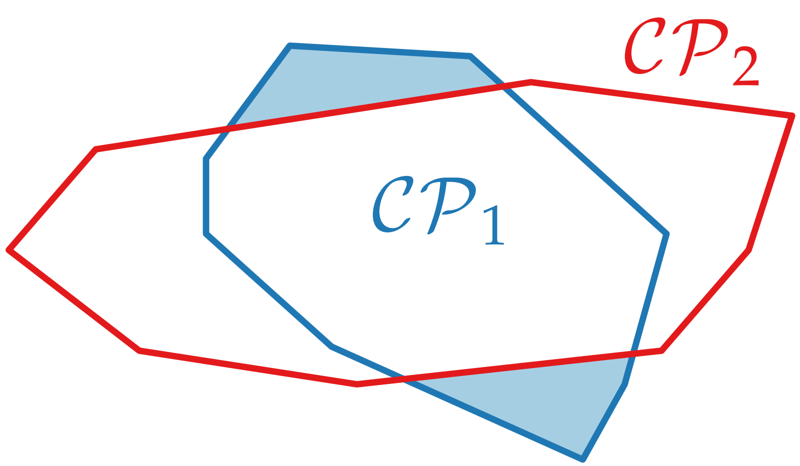
Proof. It suffices to show: $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is connected.

Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

Proof. It suffices to show: $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is connected.

Suppose $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is not connected...

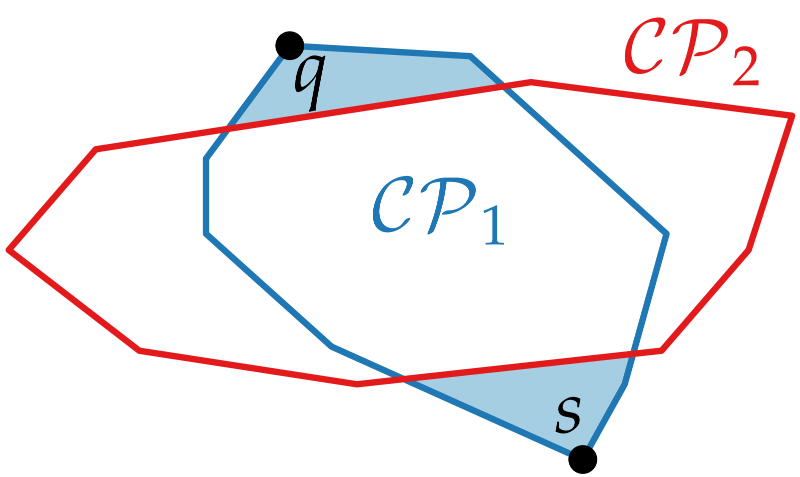


Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

Proof. It suffices to show: $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is connected.

Suppose $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is not connected...

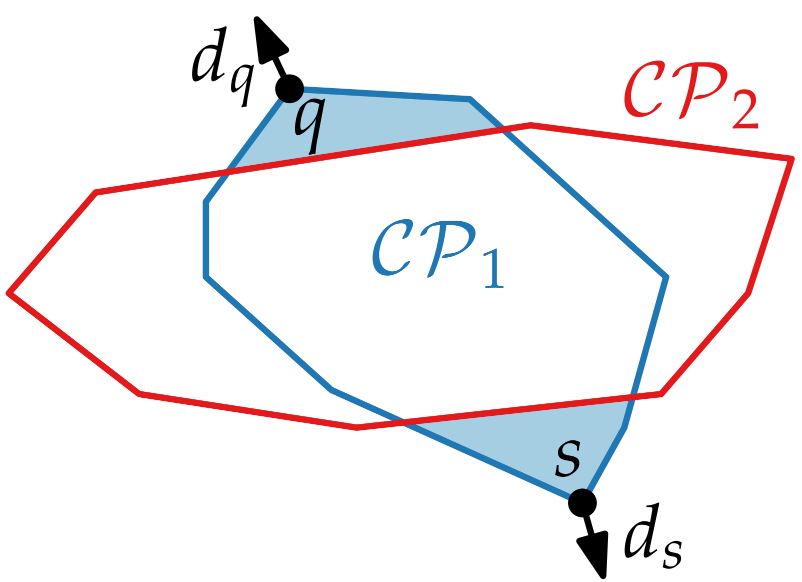


Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

Proof. It suffices to show: $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is connected.

Suppose $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is not connected...

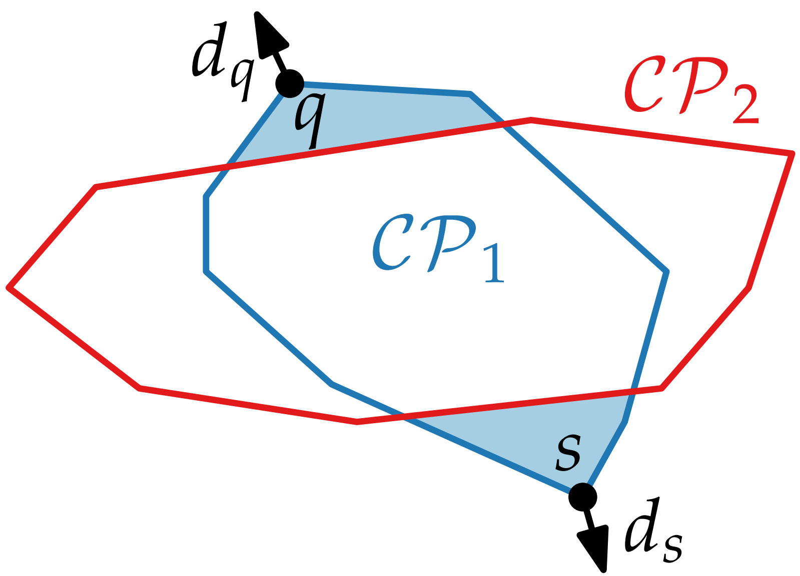


Polygonal Pseudodisks

Theorem. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are convex polygons with disjoint interiors, and $\mathcal{R}$ is another convex polygon, then $\underbrace{\mathcal{P}_1 \oplus \mathcal{R}}_{\mathcal{CP}_1}$ and $\underbrace{\mathcal{P}_2 \oplus \mathcal{R}}_{\mathcal{CP}_2}$ is a pair of pseudodisks.

Proof. It suffices to show: $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is connected.

Suppose $\mathcal{CP}_1 \setminus \mathcal{CP}_2$ is not connected...



↯ to previous observation!

Computational Geometry

Lecture 10:
Motion Planning

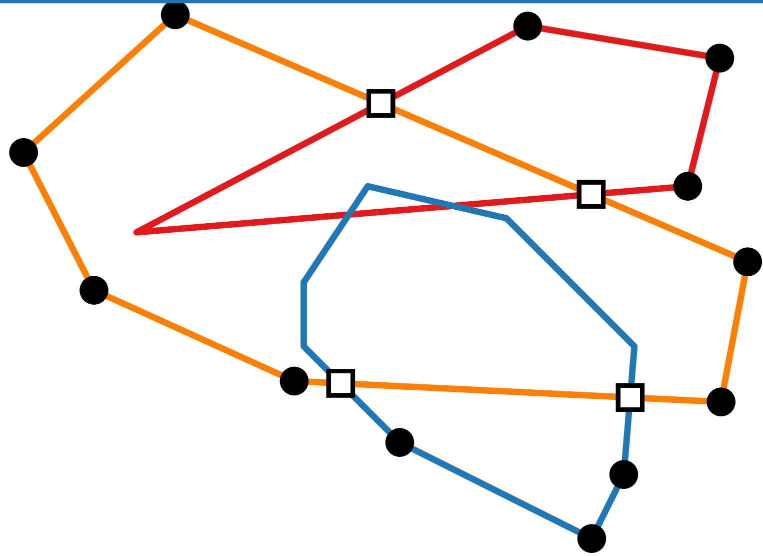
Part VI:
Union Complexity

Philipp Kindermann

Winter Semester 2020

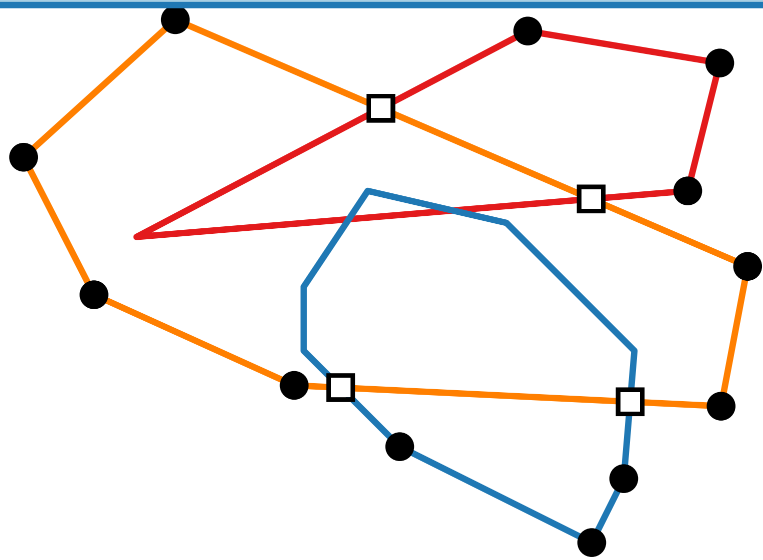
Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Union Complexity

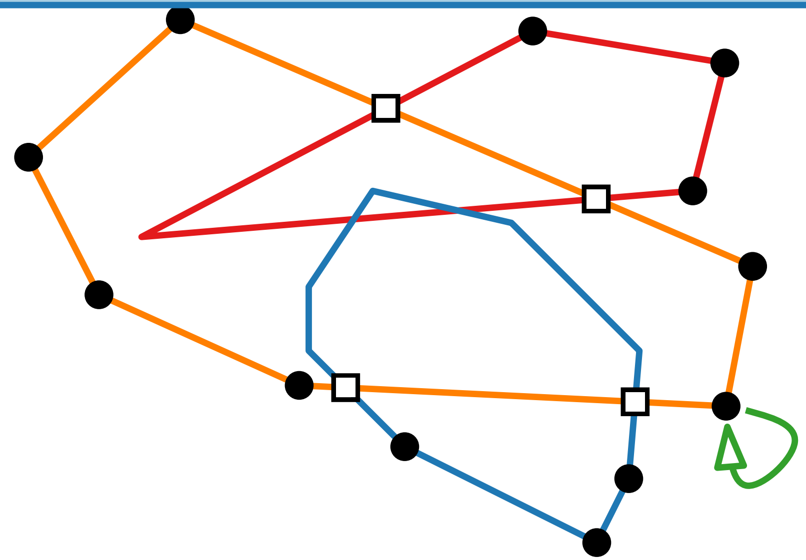
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

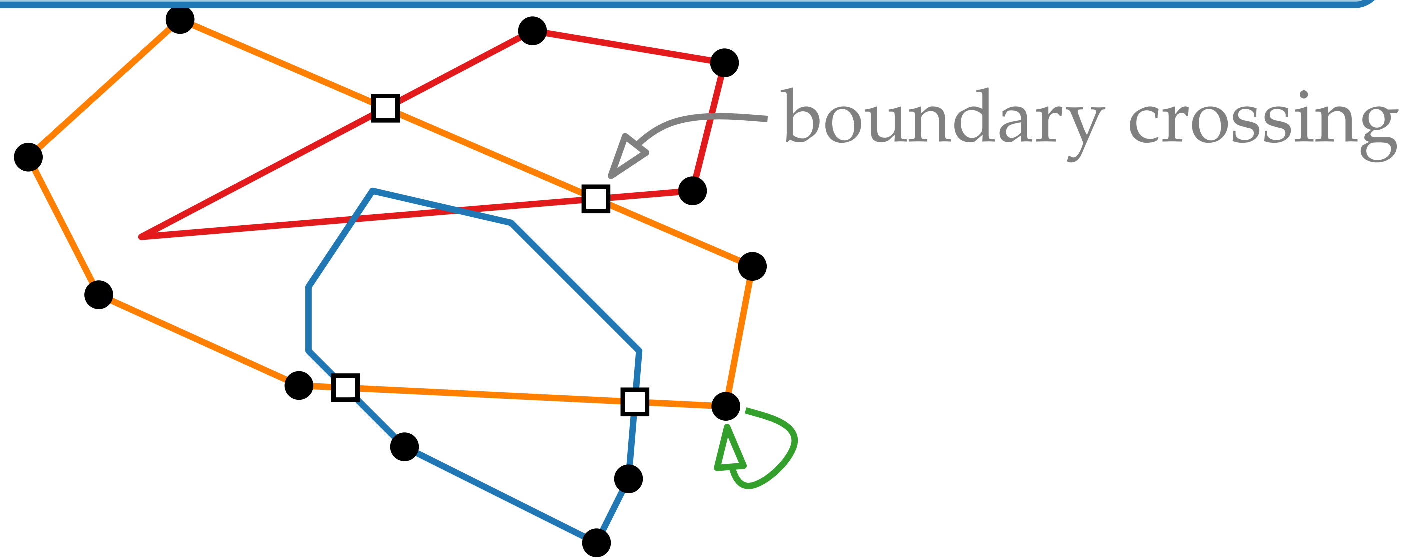
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

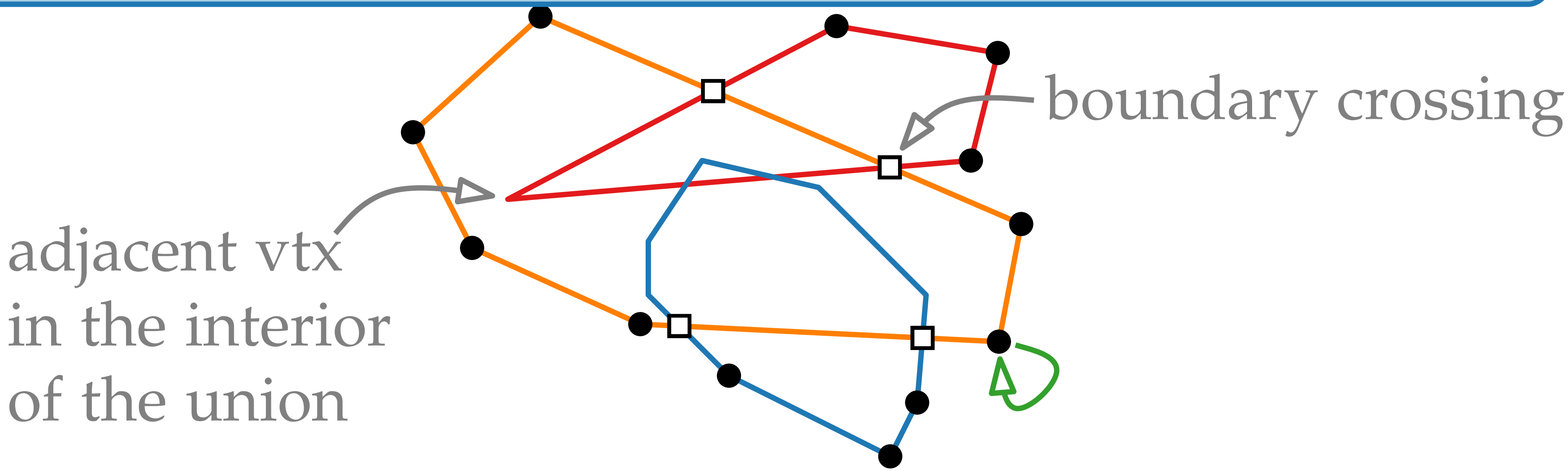
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

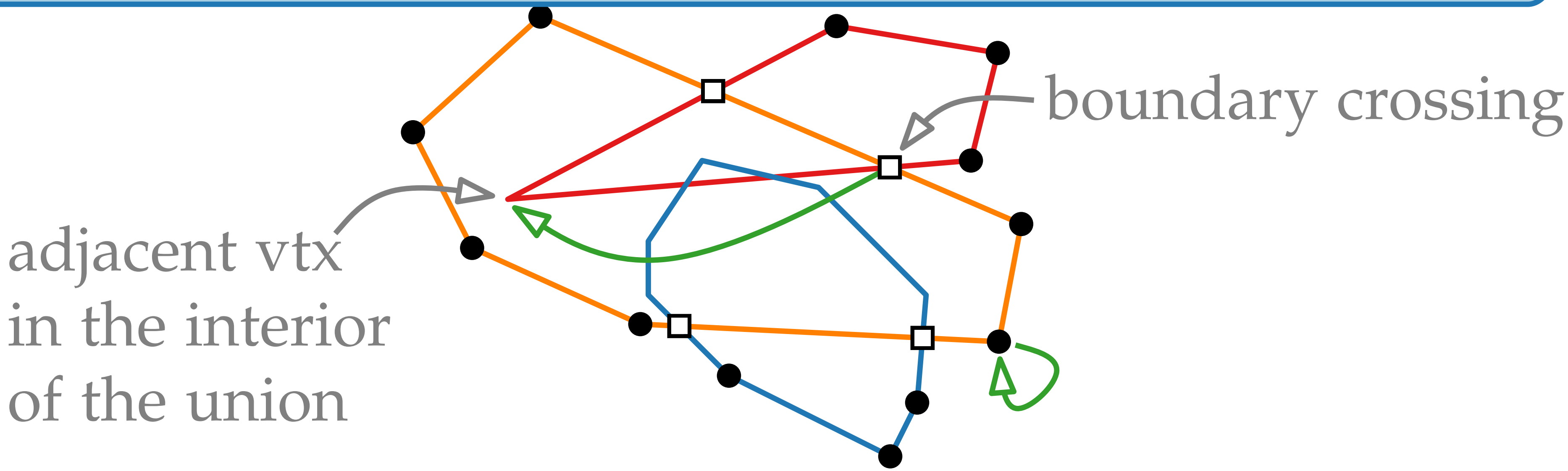
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

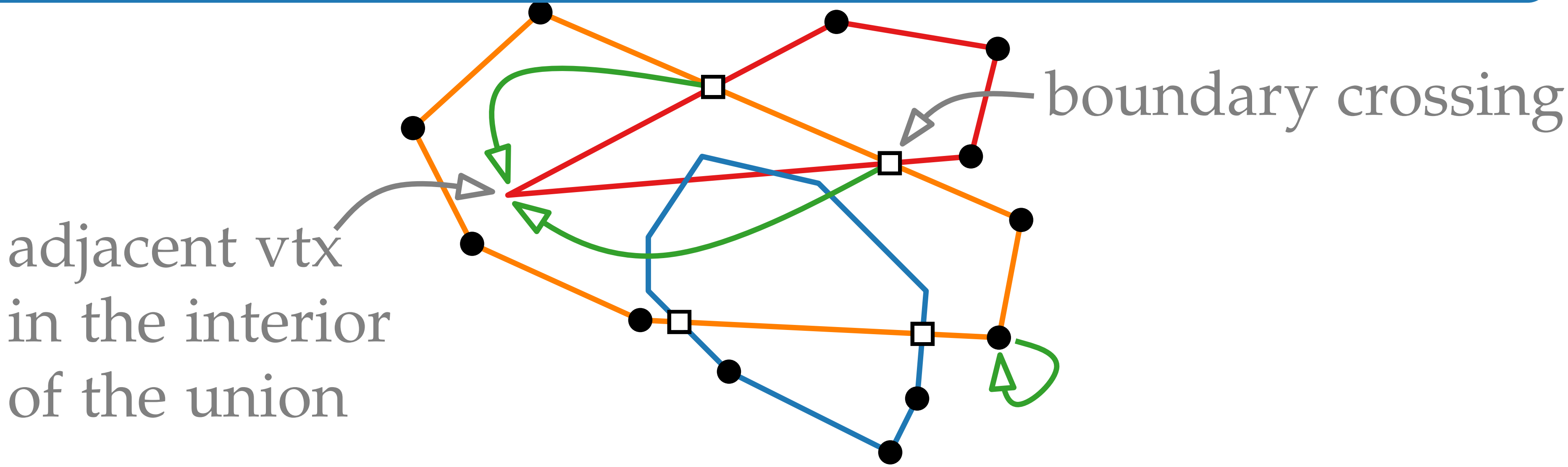
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

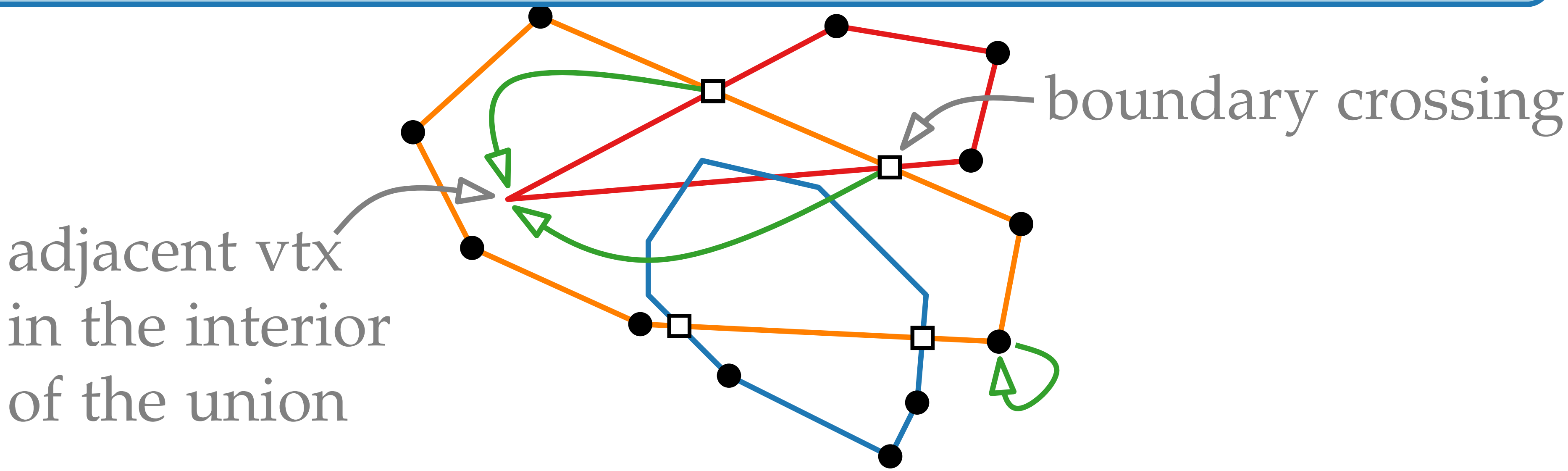
Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



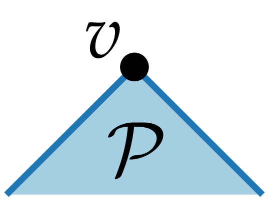
Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

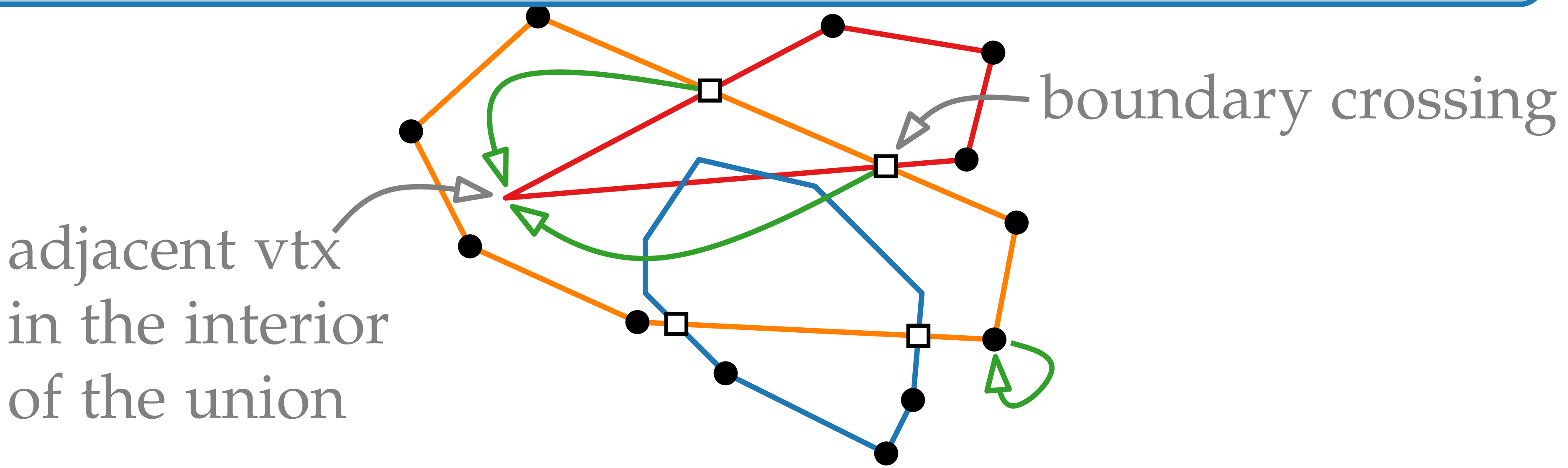


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

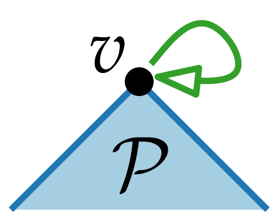


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

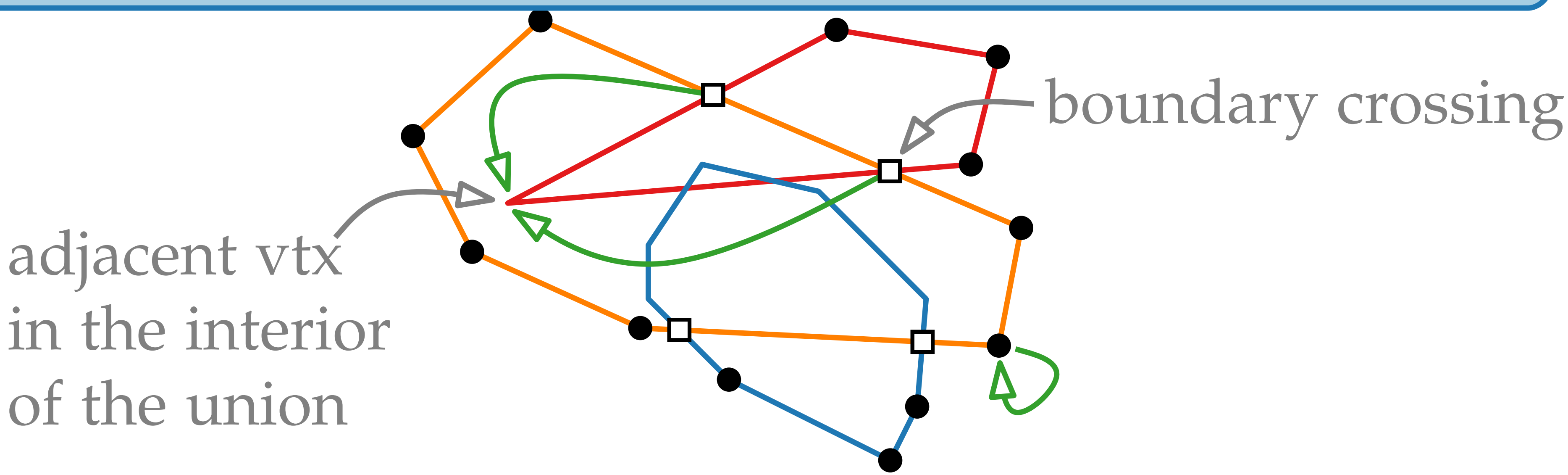


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

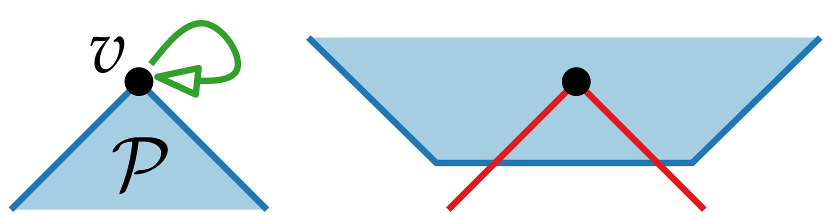


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

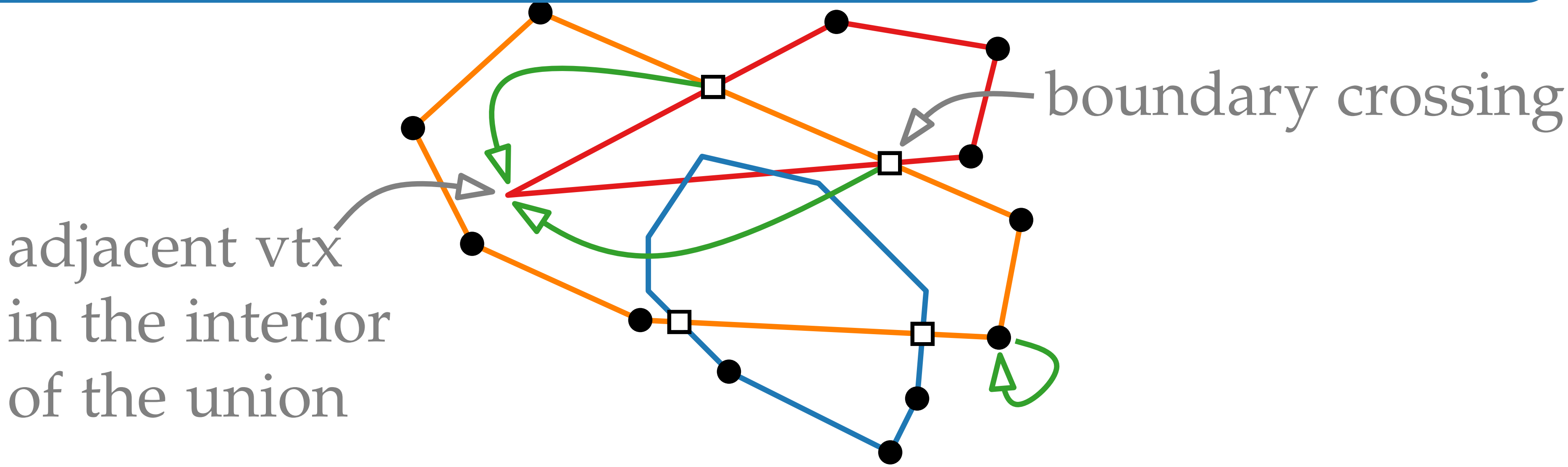


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

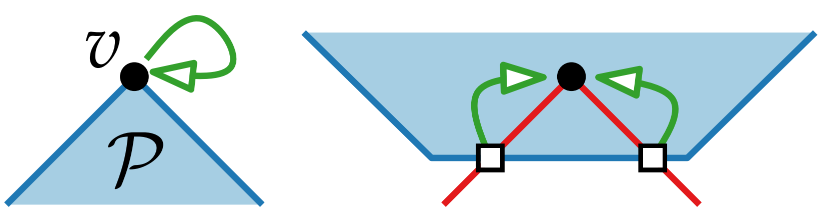


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

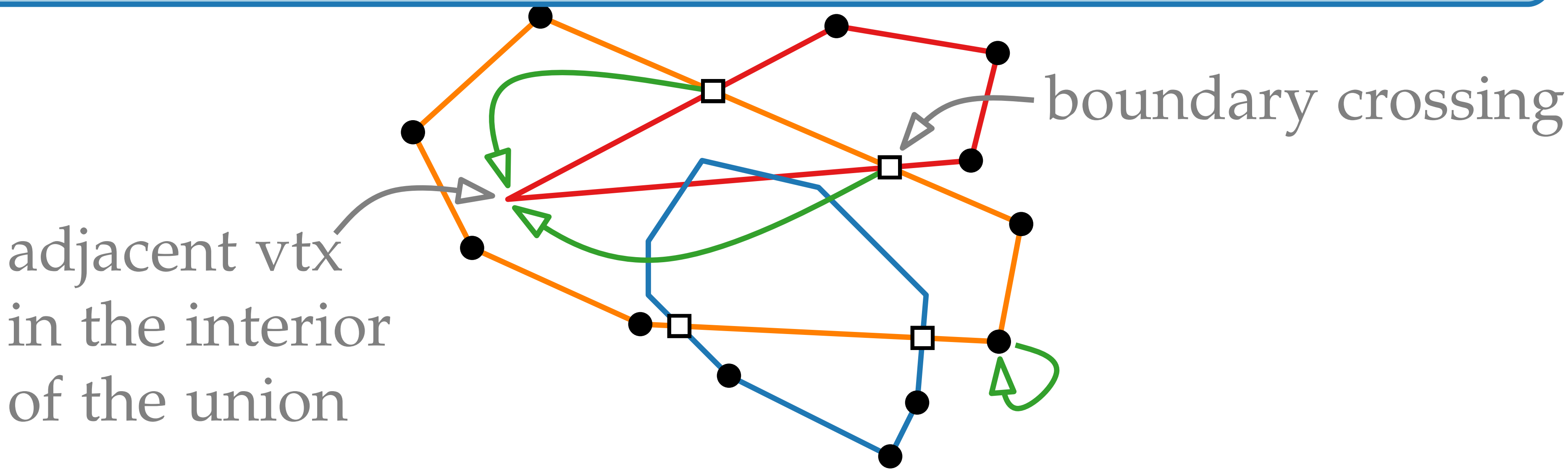


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

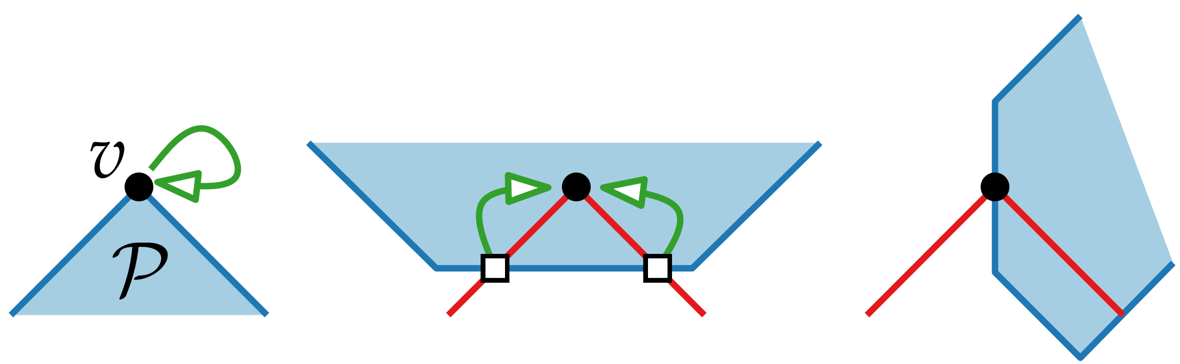


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

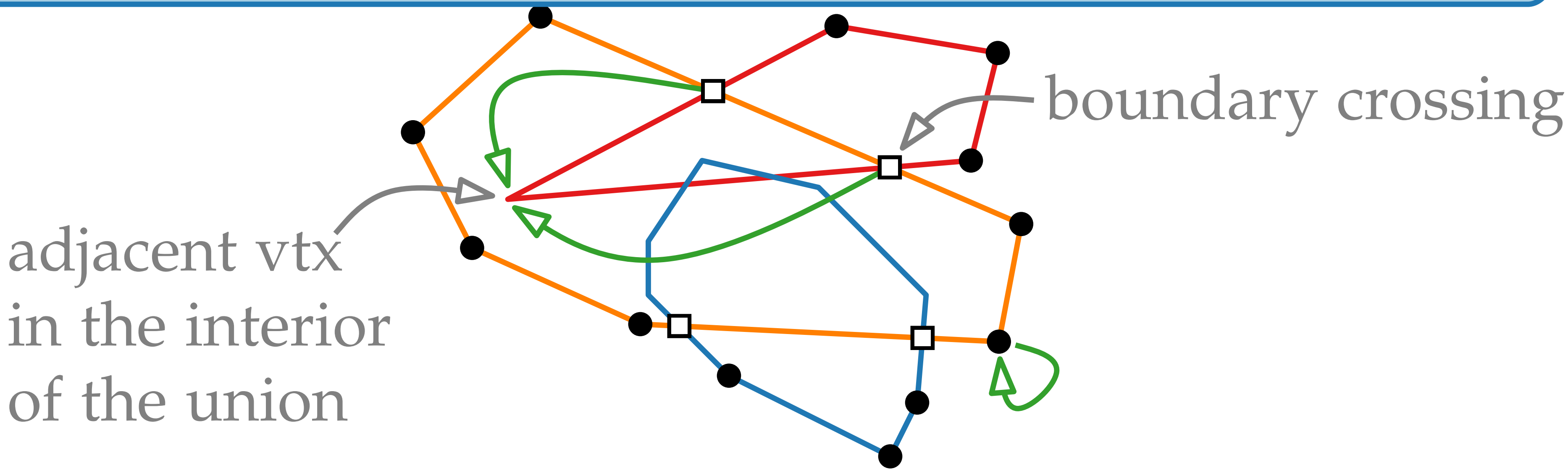


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

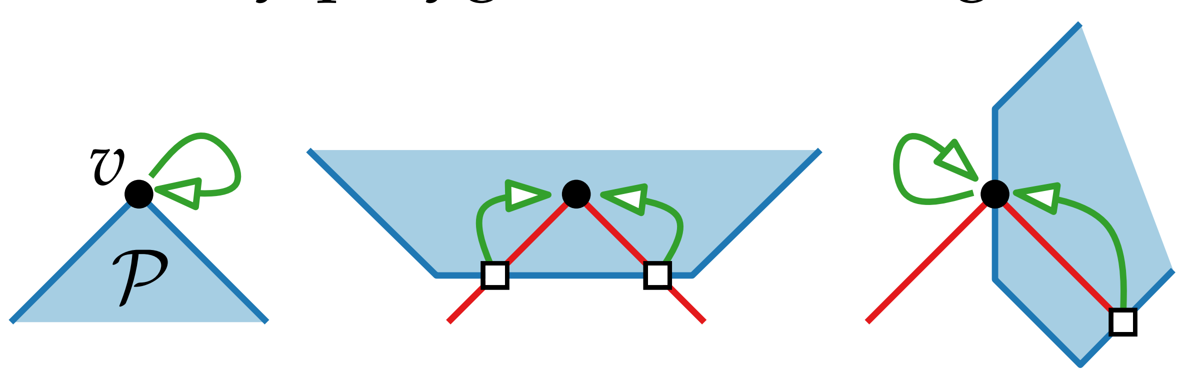


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

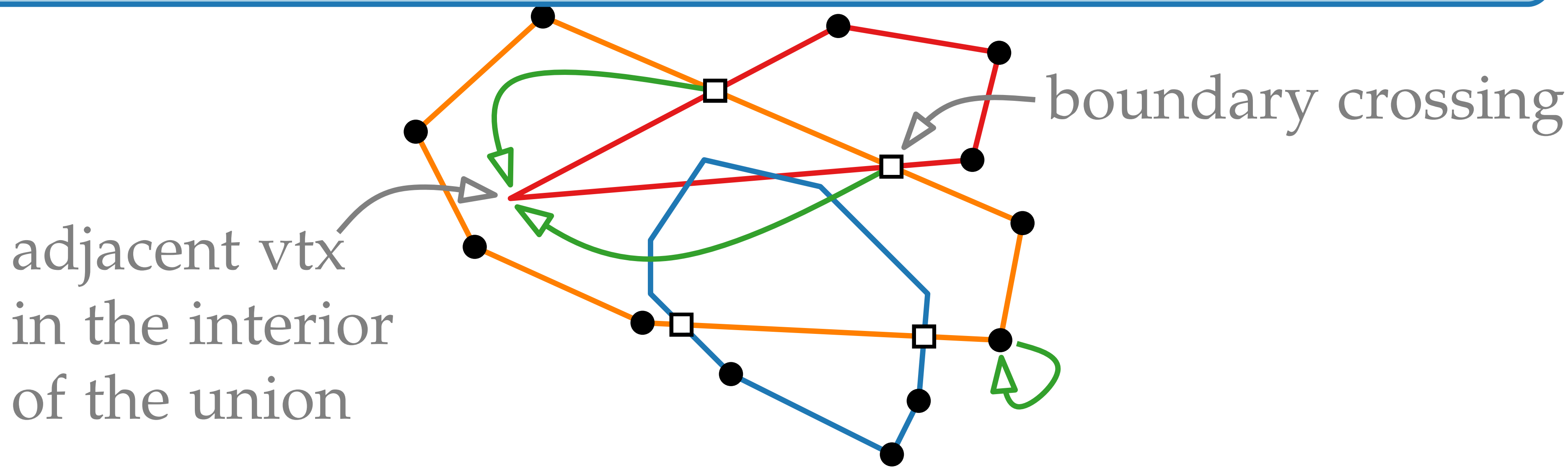


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

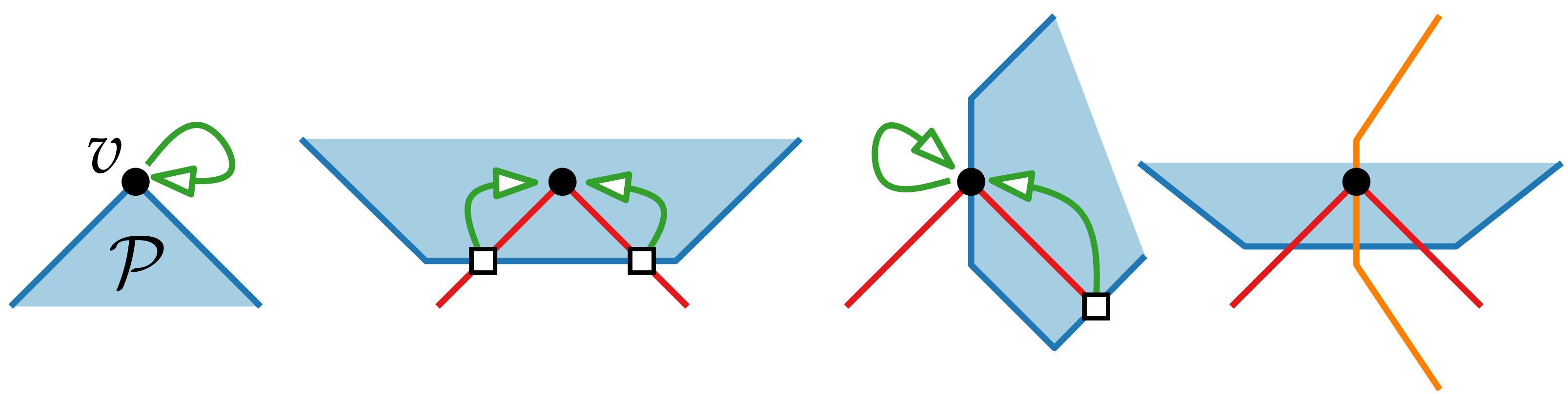


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.

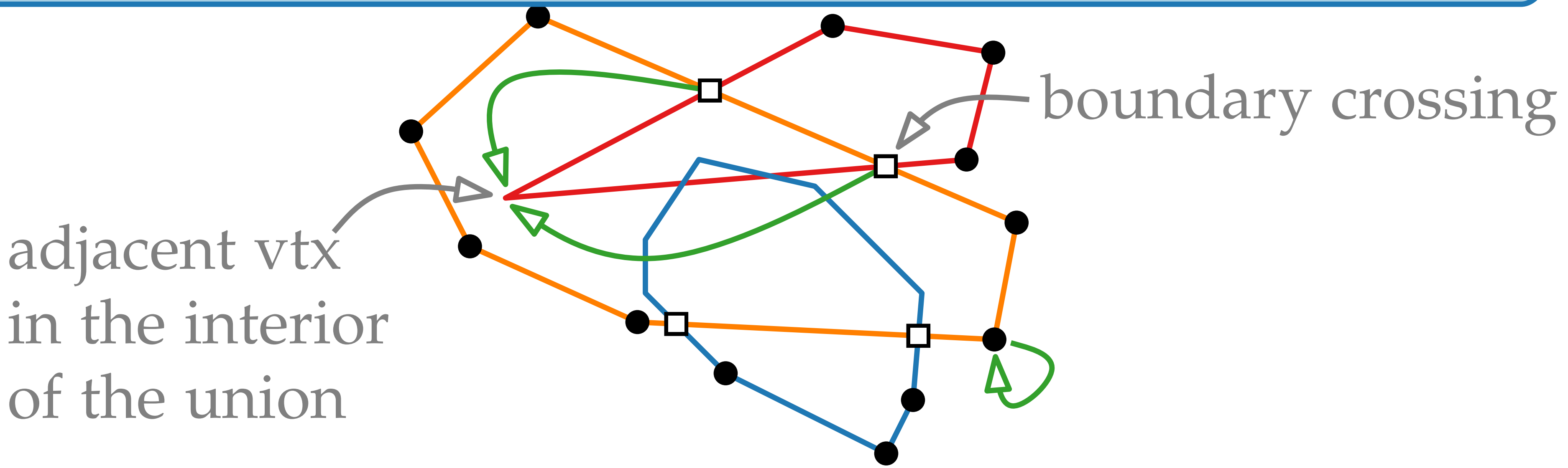


Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.

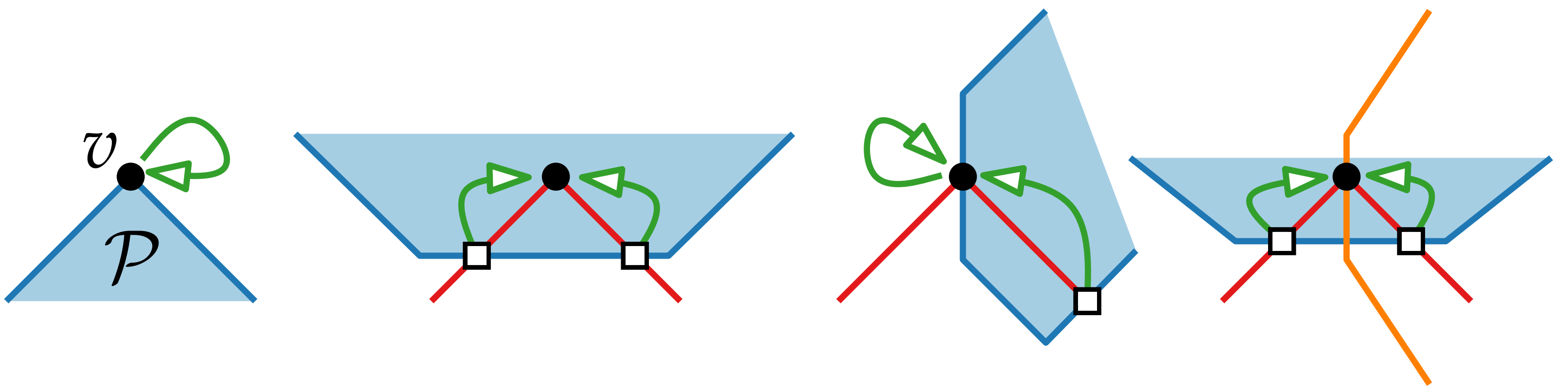


Union Complexity

Theorem. A collection S of convex polygonal pseudodisks with n vtc in total has a union with $\leq 2n$ vtc.



Proof. Charge every vtx of the union to a polygon vtx s.t. every polygon vtx is charged at most twice.



Summary and Main Result

Theorem. Let $\mathcal{R}$ be a constant-complexity convex robot, translating among a set S of disjoint polygonal obstacles with n edges in total.

Summary and Main Result

Theorem. Let $\mathcal{R}$ be a constant-complexity convex robot, translating among a set S of disjoint polygonal obstacles with n edges in total.

We can preprocess S in $O(n \log^2 n)$ time such that, given any start and goal position, we can compute in $O(n)$ time a collision-free path for $\mathcal{R}$ if it exists.

Summary and Main Result

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

Proof.

Summary and Main Result

Theorem. Let $\mathcal{R}$ be a constant-complexity convex robot, translating among a set S of disjoint polygonal obstacles with n edges in total.

We can preprocess S in $O(n \log^2 n)$ time such that, given any start and goal position, we can compute in $O(n)$ time a collision-free path for $\mathcal{R}$ if it exists.

Proof.

- Triangulate the obstacles if not convex.

Summary and Main Result

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

Proof.

- Triangulate the obstacles if not convex. Lec. 3

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3

Summary and Main Result

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

Proof.

$O(n \log n)$
- Triangulate the obstacles if not convex. Lec. 3
- Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using ...?

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using div. and conq. (merge by sweeping – Lec. 2)

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- $O(n \log^2 n)$ Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using div. and conq. (merge by sweeping – Lec. 2)

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- $O(n \log^2 n)$ Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using div. and conq. (merge by sweeping – Lec. 2)

[Argue carefully about the number of intersection pts!]

Summary and Main Result

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

Proof.

- $O(n \log n)$ Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- $O(n \log^2 n)$ Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using div. and conq. (merge by sweeping – Lec. 2)

  [Argue carefully about the number of intersection pts!]
- Find a path for a point in the complement $\mathcal{C}_{\text{free}}$.

Summary and Main Result

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

Proof.

- $O(n \log n)$ — Triangulate the obstacles if not convex. Lec. 3
- $O(n)$ — Compute $\mathcal{CP}_i$ for every convex obstacle $\mathcal{P}_i$.
- $O(n \log^2 n)$ — Compute their union $\mathcal{C}_{\text{forb}} = \bigcup_i \mathcal{CP}_i$ using div. and conq. (merge by sweeping – Lec. 2)

  [Argue carefully about the number of intersection pts!]
- $O(n)$ — Find a path for a point in the complement $\mathcal{C}_{\text{free}}$.