Introduction to Robot Motion Planning & Navigation
Module 4

Solmaz S. Kia (solmaz.eng.uci.edu)

solmaz@uci.edu

Mechanical and Aerospace Engineering Department

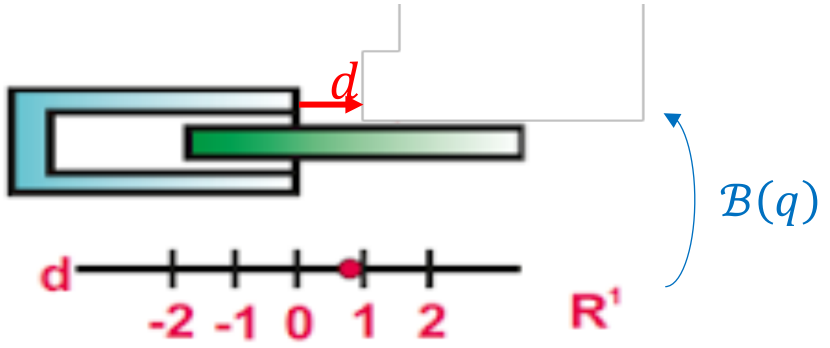
University of California Irvine

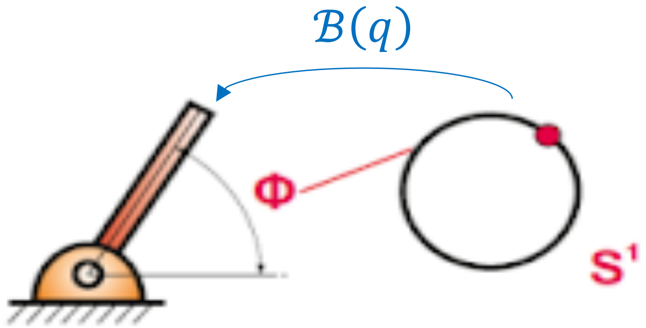
Chapter 4: Free Configuration Spaces via Sampling and Collision Detection

- represent obstacles and the free space when the robot is composed of a single or multiple rigid bodies with proper shape, position and orientation,
- compute free configuration spaces via sampling and collision detection,
- discuss sampling methods, and
- discuss collision detection methods.

Configuration Space

- A configuration of a robot is a minimal set of variables that specifies the position and orientation of each rigid body composing the robot. The robot configuration is usually denoted by the letter q.



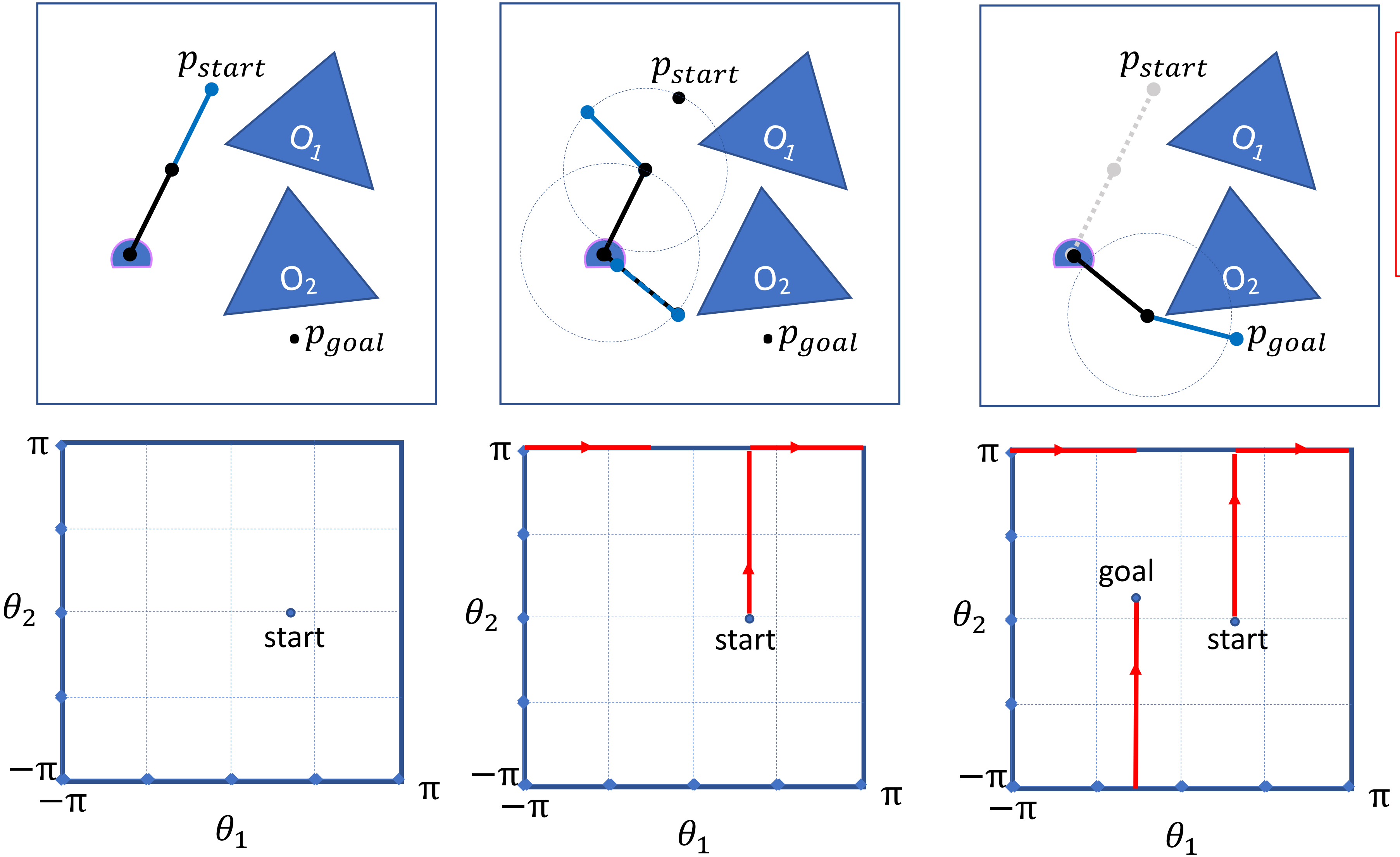


- The configuration space is the set of all possible configurations of a robot, denoted by the letter Q, so that $q \in Q$.

- The number of degrees of freedom of a robot is the dimension of the configuration space, i.e., the minimum number of variables required to fully specify the position and orientation of each rigid body belonging to the robot.

- The configuration map, and it maps each point $q \in Q$ to the set of all points $\mathcal{B}(q)$ of the workspace belonging to the robot.

Motion Planning for Robots with Finite Shape and Size



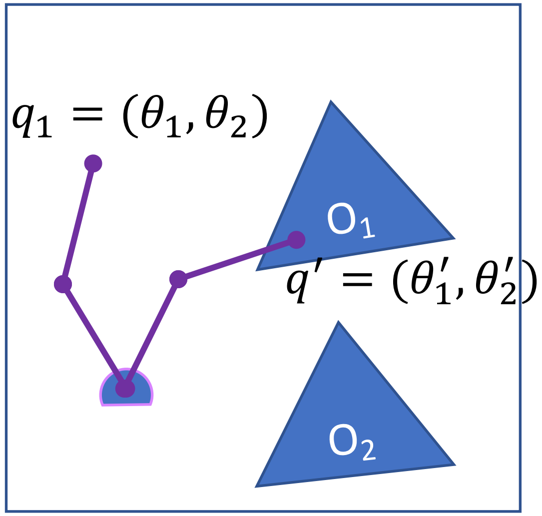
- A workspace $W \subset R^2$;
- Some obstacles $O_1, O_2, \cdots, O_n$;
- $W_{free} = W \backslash (O_1 \cup O_2 \cup \cdots \cup O_n)$

- A configuration space;
- …..
- $Q_{free} = \cdots$

Motion planning for rigid body robots: given a motion planning problem in the workspace "move from point $p_{start} \in W$ to point $P_{goal} \in W$," we need to translate this specification into the configuration space, i.e., move from a configuration $q_{start} \in Q$ to a configuration $q_{goal} \in Q$.

Free Configuration Space

The key problem: what robot configurations correspond to feasible positions of the robot, i.e., configurations in Q such that the robot is not in collision with any obstacle in W.



The free configuration space Q_{free} is the set of configurations q such that all points of the robot are inside W_{free}:

$$Q_{free} = \{q \in Q | \mathcal{B}(q) \in W_{free}\}$$

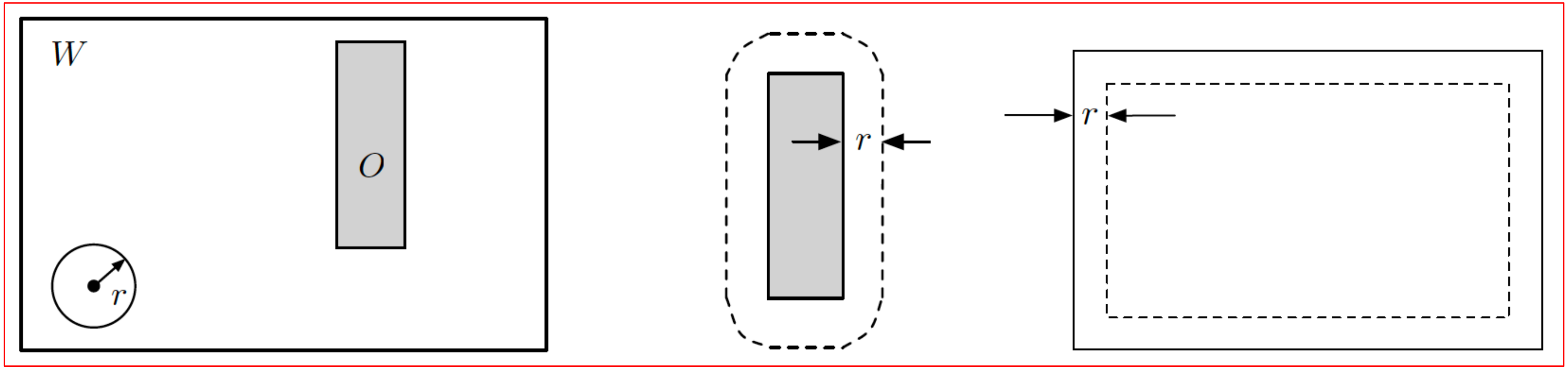
Given an obstacle O in workspace, the corresponding configuration space obstacle O_Q is the set of configurations q such that the robot at configuration q is in collision with the obstacle Q:

$$O_Q = \{q \in Q | \mathcal{B}(q) \; overlaps \; with \; O\}$$

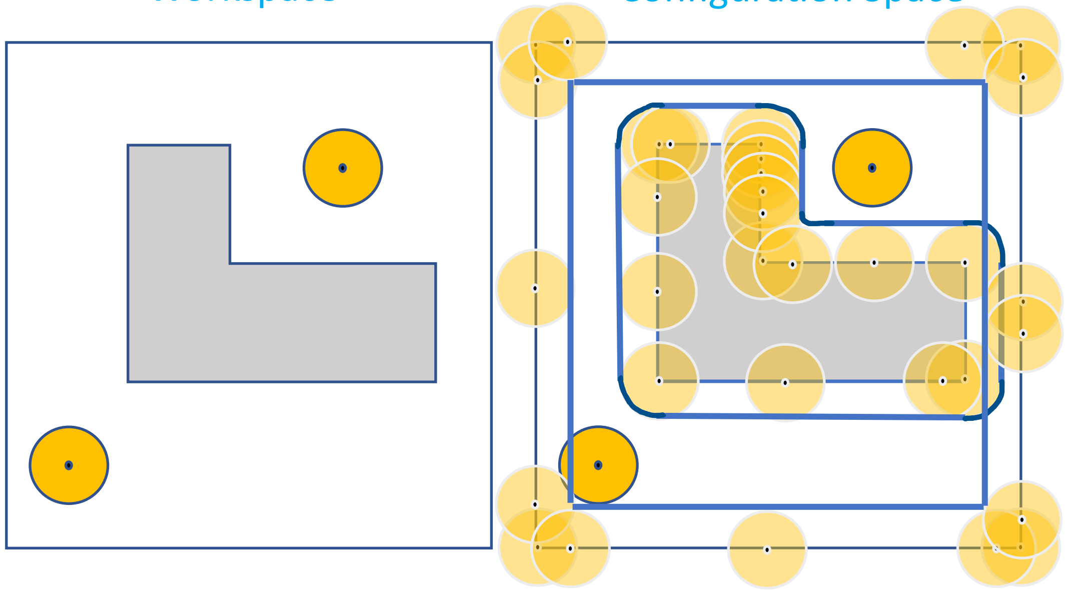
- A workspace $W \subset R^2$;
- Some obstacles $O_1, O_2, \cdots, O_n$;
- $W_{free} = W \backslash (O_1 \cup O_2 \cup \cdots \cup O_n)$

- A configuration space;
- Some configuration space obstacles $O_{Q1}, O_{Q2}, \cdots, O_{Qn}$;
- $Q_{free} = Q \backslash (O_{Q1} \cup O_{Q2} \cup \cdots O_{Qn})$

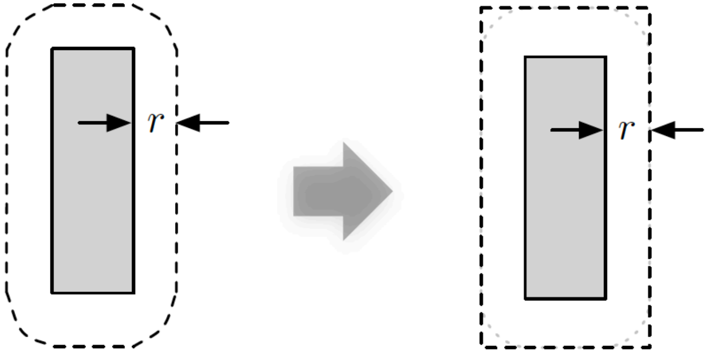
Free Configuration Space for the Disk Robot



Workspace Configuration Space

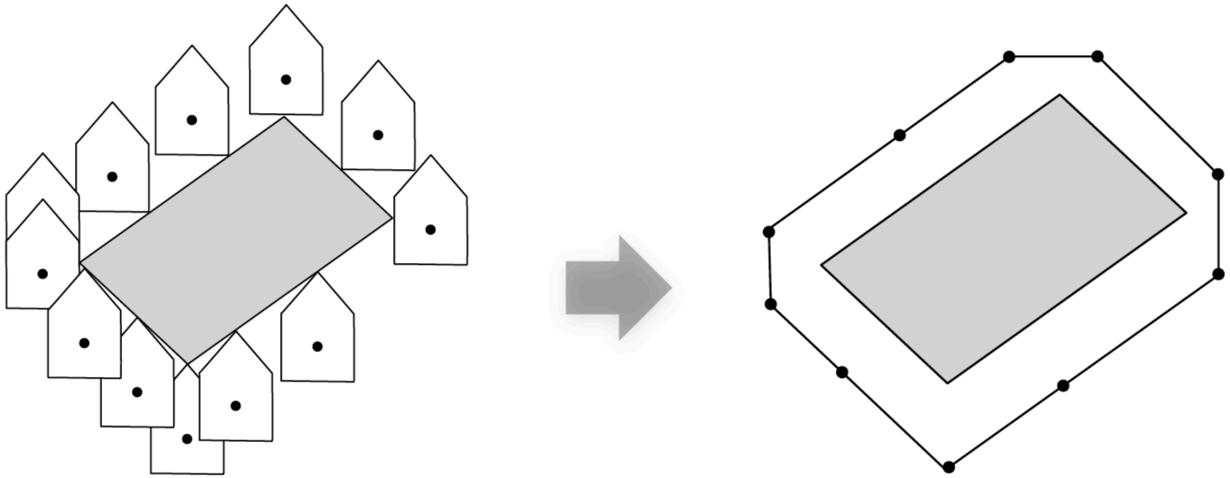


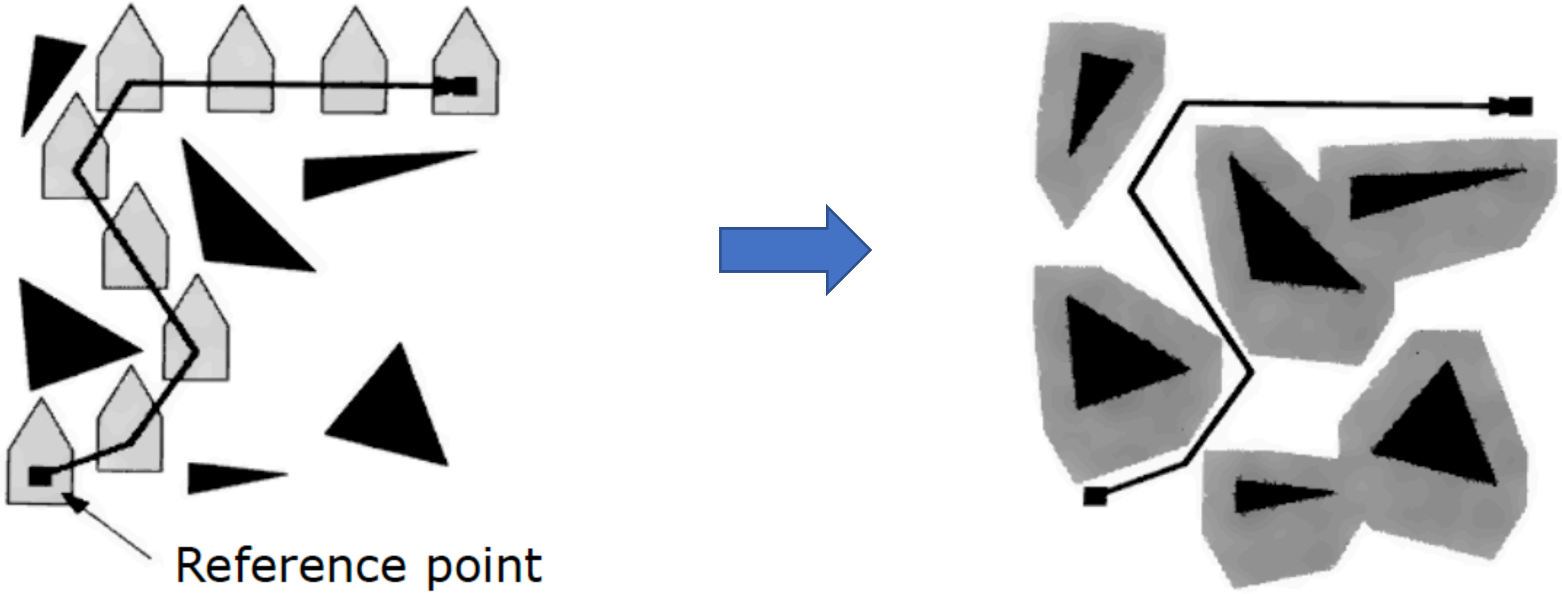
Approximating expanded obstacles by a larger polygonal obstacle



Free Configuration Space for the Translating Polygon Robot

- Q-space is obtained by sliding the robot along the edge of the obstacle regions



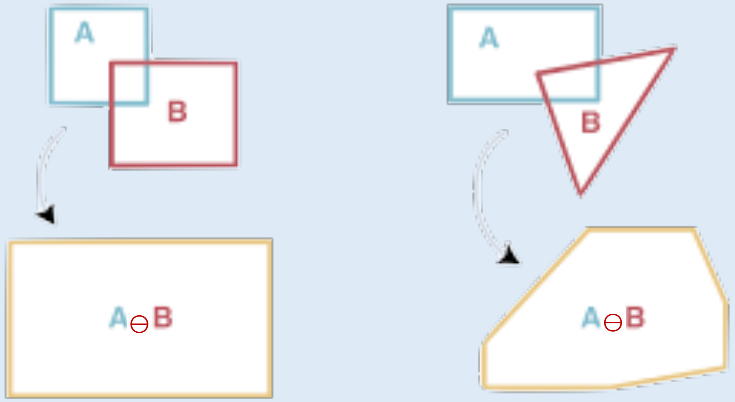


Some Mathematical Definitions

Given two sets $\mathcal{S}_1 \subset R^2$ and $\mathcal{S}_2 \subset R^2$, the Minkowski difference ((or geometric difference) $\mathcal{S}_1 \ominus \mathcal{S}_2$ is defined by

$$\mathcal{S}_1 \ominus \mathcal{S}_2 = \{p - q | p \in \mathcal{S}_1 \;\; and \;\; q \in \mathcal{S}_2\}$$

That is, Minkowski difference $\mathcal{S}_1 \ominus \mathcal{S}_2$ is the result of subtracting every point in $\mathcal{S}_1$ from every point in $\mathcal{S}_2$.

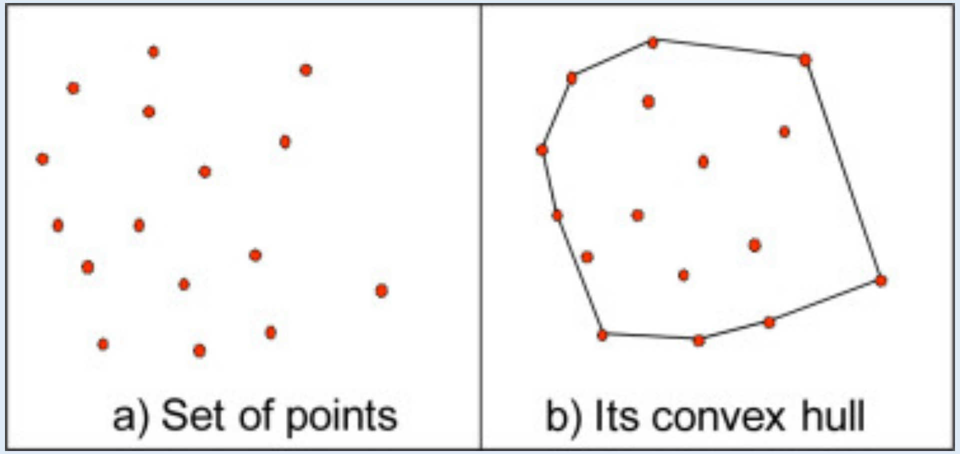


Convex hull of a group of points as the minimum-perimeter convex set containing them.

- The convex hull of a group of points is a convex polygon.

- Graphically, the convex hull can be obtained by snapping a tight rubber band around all the points (the length of the rubber band is the perimeter of the envelope).

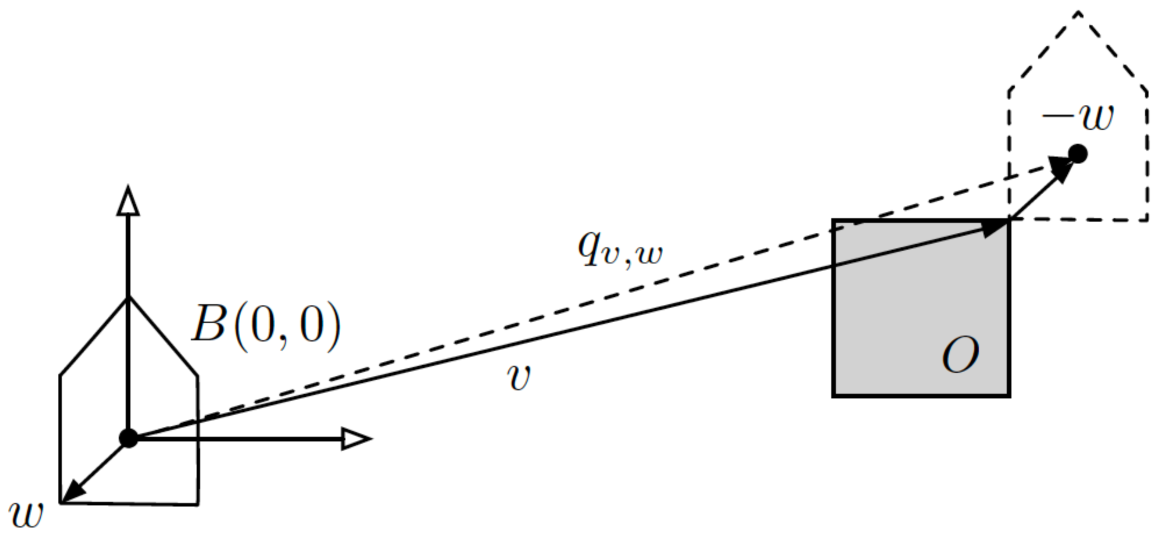
- Every convex polygon is the convex hull of its vertices.



Polygonal Obstacles: from Workspace to Configuration Space

Proposition 4.1 Assume the robot body, with reference position $\mathcal{B}(0,0)$ and the obstacle O are convex polygons with n and m vertices respectively. Then the resulting configuration space obstacle O_Q is a convex polygon with at most $n + m$ vertices and satisfies

$$O_Q = O \ominus \mathcal{B}(0,0)$$

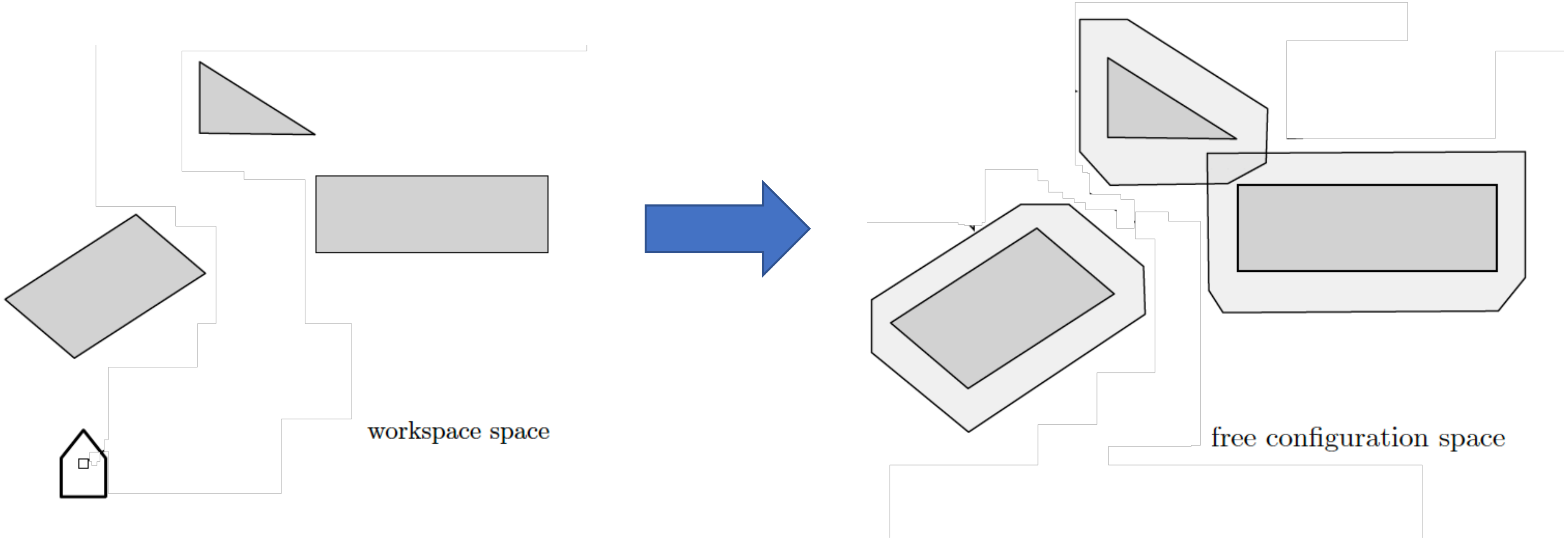


Polygonal Obstacles: from Workspace to Configuration Space

Minkowski-difference-via-convex-hull algorithm

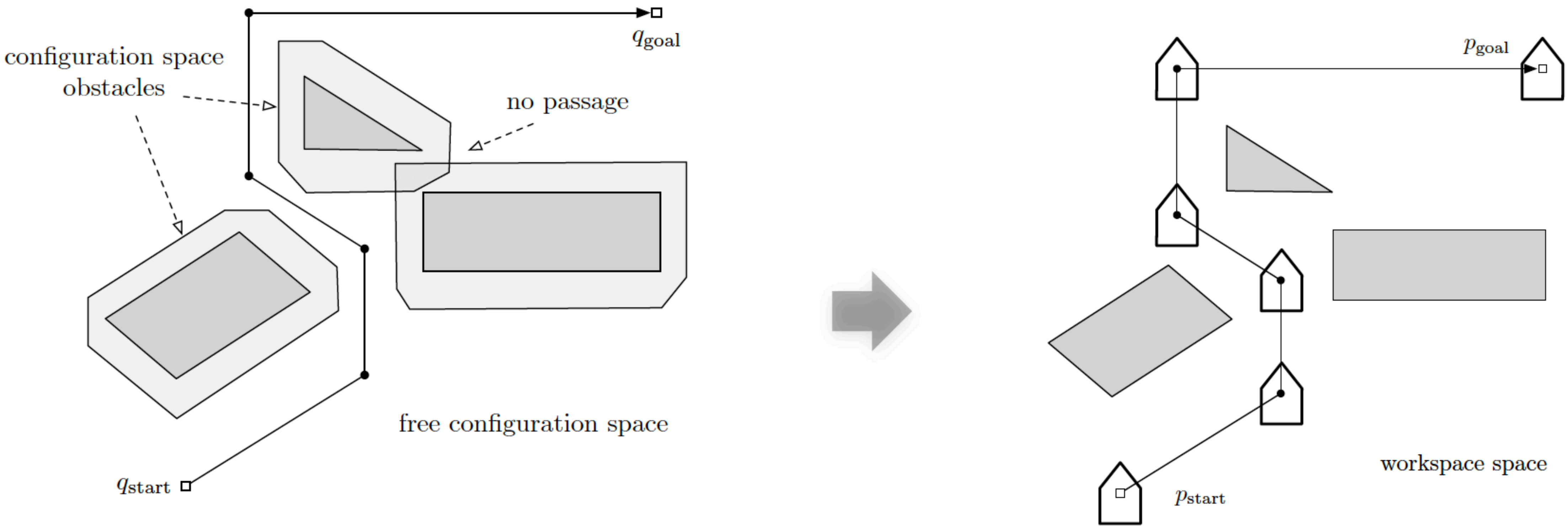
Input: two convex polygonal subsets P_1 and P_2 of $\mathbb{R}^2$
Output: the Minkowski difference $P_1 \ominus P_2$

```
for each vertex v of P1 :
    for each vertex w of P2 :
        compute the difference point v − w
return the convex hull of all difference points
```

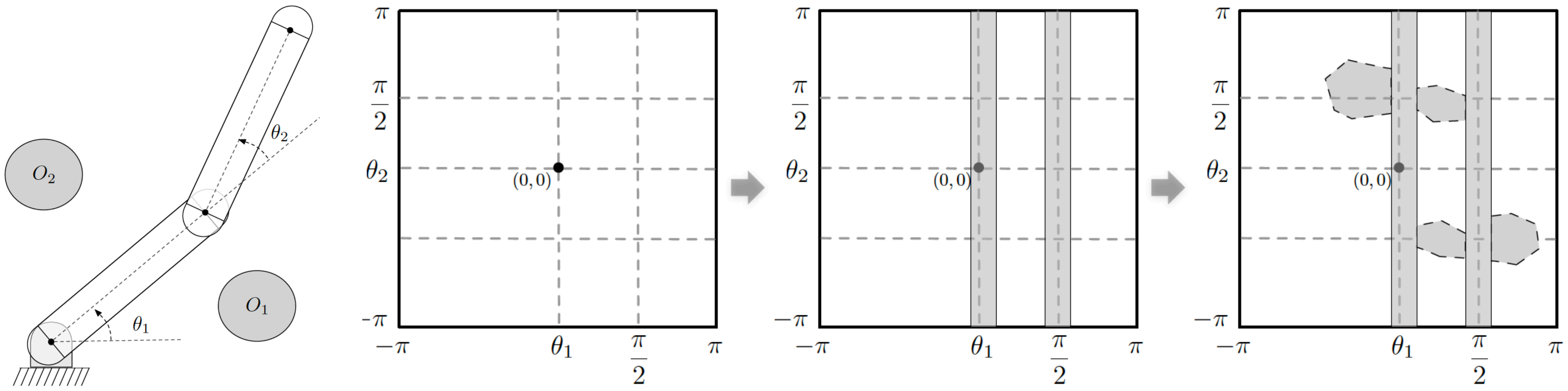


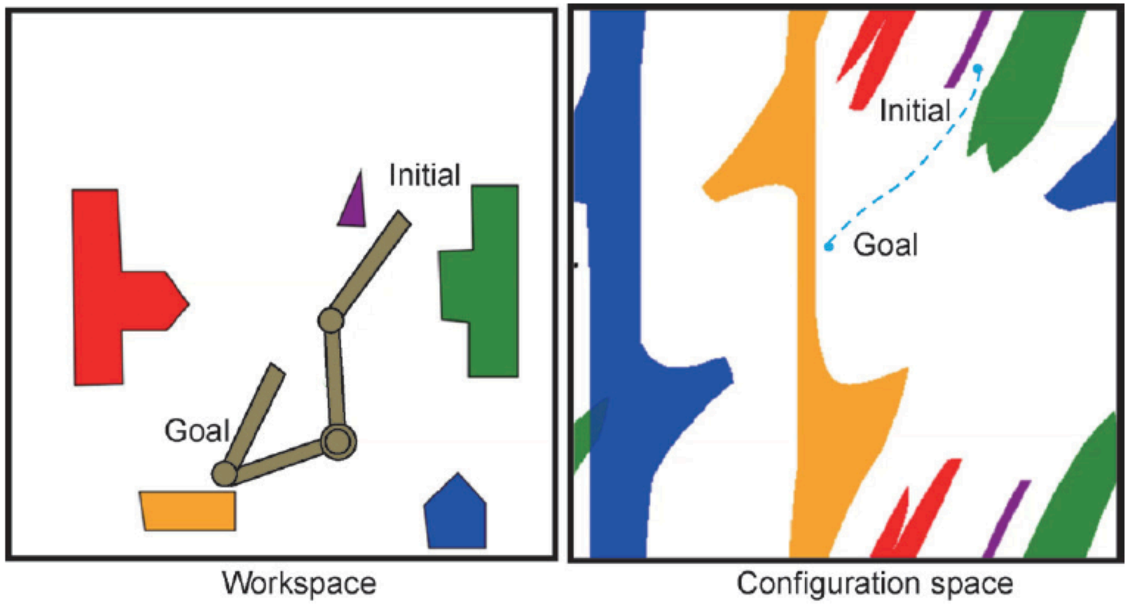
Motion planning for Robots with Finite Shape and Size

Motion planning for rigid body robots: given a motion planning problem in the workspace "move from point $p_{start} \in W$ to point $P_{goal} \in W$," we need to translate this specification into the configuration space, i.e., move from a configuration $q_{start} \in Q$ to a configuration $q_{goal} \in Q$.



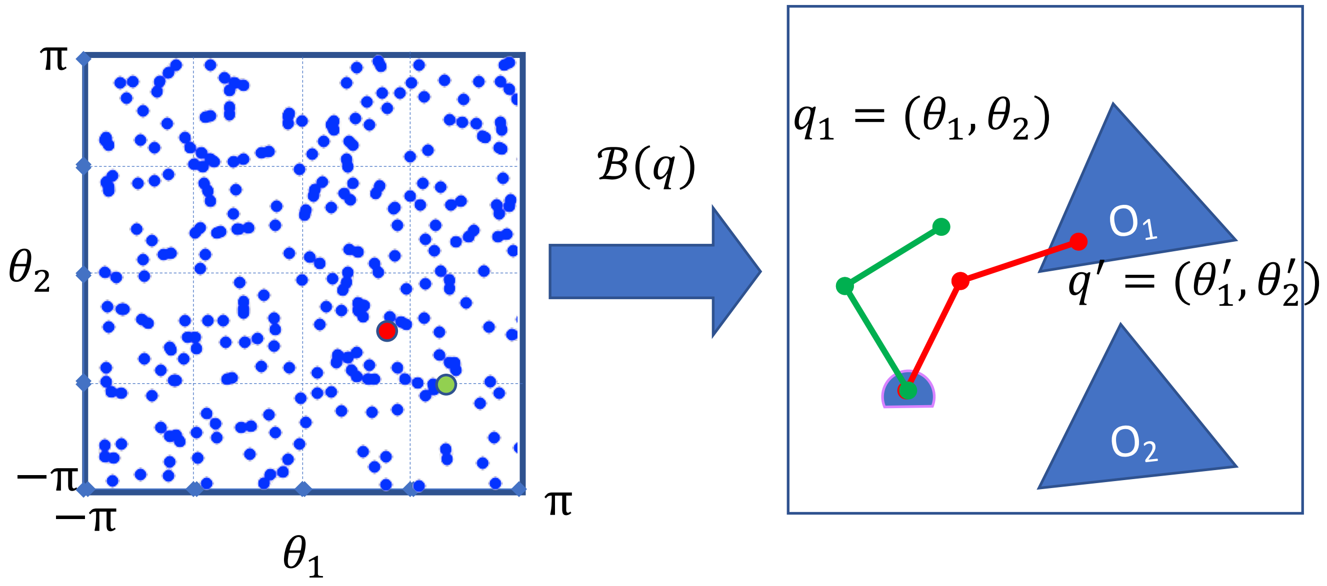
Free Configuration Space for the 2-link Robot





Jia Pan, Dinesh Manocha, "Efficient Configuration Space Construction and Optimization for Motion Planning"
Engineering 2015, 1(1): 46–57
DOI 10.15302/J-ENG-2015009

Numerical Computation of the Free Configuration Space

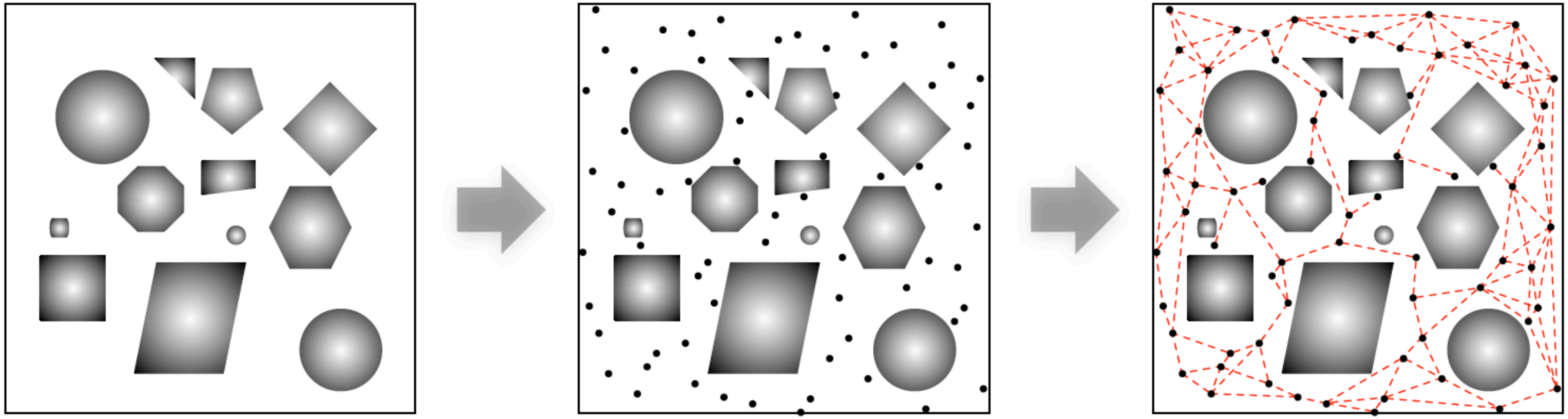


sampling & collision-detection algorithm

Input: A number of samples n, the free workspace W_{free}, and the robot configuration map $B(q)$
Output: A set of configurations in the free configuration space

1: Initialize `free-configs` $:= \emptyset$
2: Compute a sequence of sample configurations $q_1, \ldots, q_n$
3: **for** each configuration sample q_i in the sequence :
4: compute the positions of the robot rigid bodies corresponding to the sample, $B(q_i)$
5: detect if the robot collides with the obstacles (i.e., test if $B(q_i) \subset W_{\text{free}}$)
6: **if** robot does not collide with obstacles and is inside the workspace :
7: Add q_i to `free-configs`
8: **return** `free-configs`

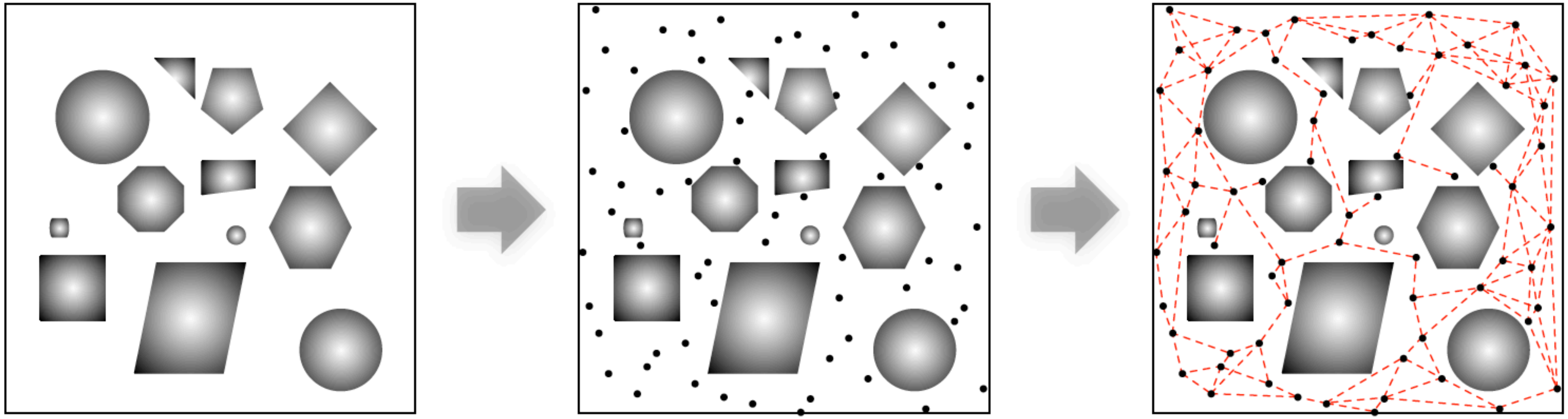
Numerical Sampling of Configuration Space→ Motion Planning



"cloud representation" of the free configuration space for robots and obstacles of arbitrary shape

compute a roadmap from a cloud in the next chapter

Numerical Sampling of Configuration Space



A sampling method should have certain properties:

- **Uniformity**: the samples should provide a "good covering" of space. Mathematically, this can be formulated using the notion of dispersion.

- **Incremental property**: the sequence of samples should provide good coverage at any number n of samples. In other words, it should be possible to increase n continuously and not only in discrete large amounts.

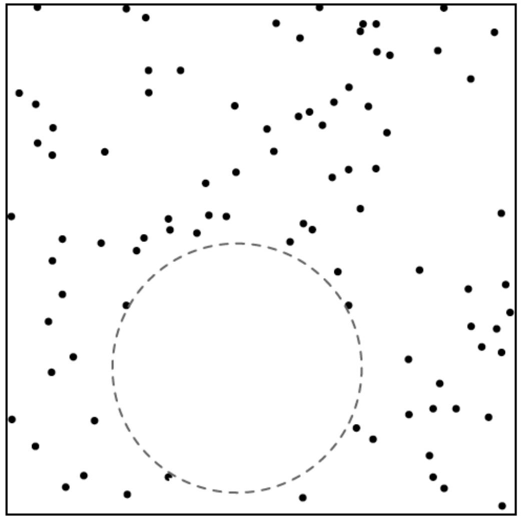
- **Lattice structure**: given a sample, the location of nearby samples should be computationally easy to determine.

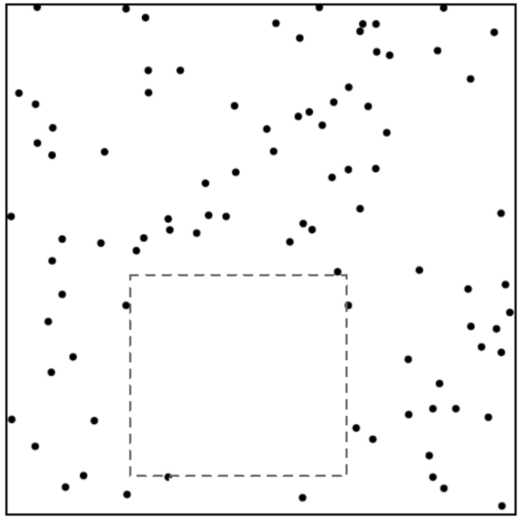
Numerical Sampling: Dispersion

Consider the d-dimensional unit cube $X = [0,1]^d \subset \mathbb{R}^d$. The *sphere-dispersion* and the *square-dispersion* of a set of points P in the set X are defined by (see also Figure 4.13)

$$\text{dispersion}_{\text{sphere}}(P) = \text{radius of largest empty } d\text{-sphere with center in } X,$$

$$\text{dispersion}_{\text{square}}(P) = \frac{1}{2}(\text{side length of largest empty } d\text{-dimensional cube with center in } X)$$





Given a space X,and set of points $P \subset X$, and a metric distance, the dispersion of P with respect to the distance metric is defined as

$$\text{dispersion}(P) = \max_{x\in X} \min_{p\in P} \text{dist}(x,p).$$

That is, the dispersion is defined as maximum distance from a point in X to its nearest sample in P.

(i) sphere dispersion uses the 2-norm:

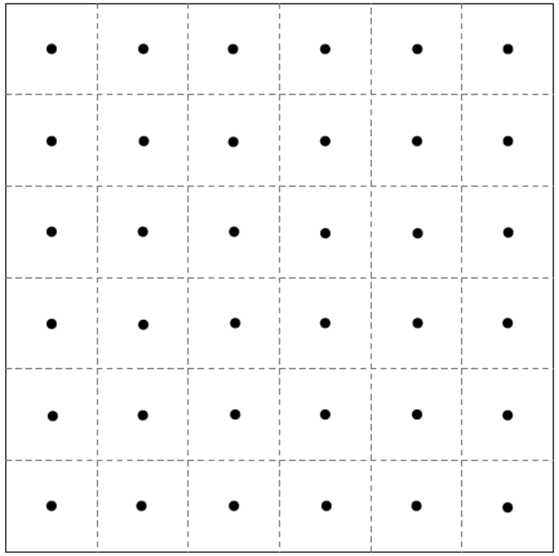
$$\text{dist}(x,p) = \|x-p\|_2 = \sqrt{(x_1-p_1)^2 + \cdots + (x_d-p_d)^2},$$

(ii) square dispersion uses the ∞-norm:

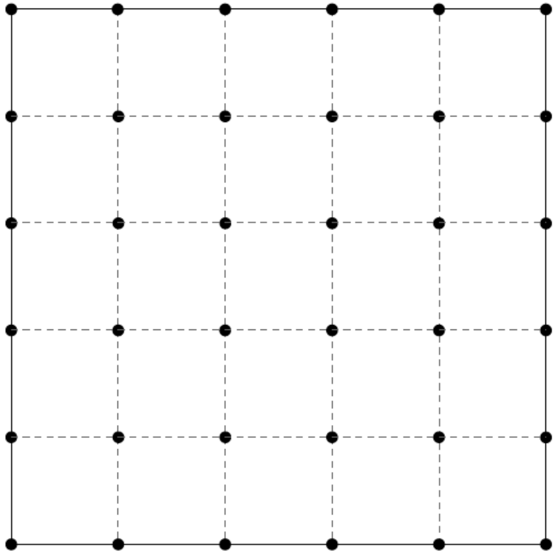
$$\text{dist}(x,p) = \|x-p\|_\infty = \max(|x_1-p_1|, \ldots, |x_d-p_d|),$$

Numerical Sampling: Uniform Grids

Uniform grids: There are two ways of defining uniform grids in the unit cube $X = [0,1]^d$: center grid and the corner grid. Both grids with n points can be defined if $n = k^d$ for some number k.



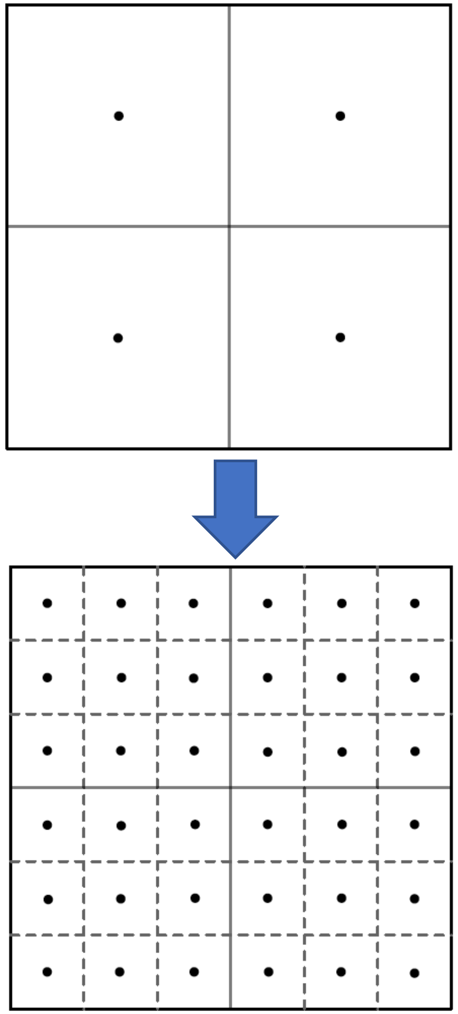
uniform center grid (Sukharev grid)



Corner grid

$$\text{dispersion}_{\text{square}}(P_{\text{center grid}}(n,d)) = \frac{1}{2\sqrt[d]{n}}.$$

multi-resolution versions of the uniform center grid



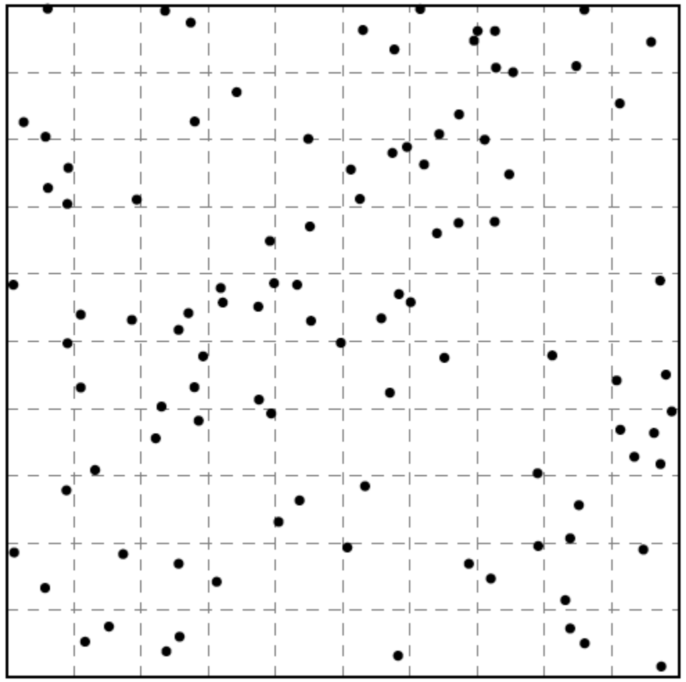
A step increase in resolution correspond to a jump in number of samples from n to $n3^d$.

Incremental property 👎

Numerical sampling: Random Sampling

Random and pseudo-random sampling Adopting a random number generator is usually a very simple approach to (uniformly or possibly non-uniformly) sample the cube $X = [0,1]^d$. An example set of 100 randomly generated points is shown in Figure 4.16.

Note: Let P be a set of n points generated independently and uniformly over X. As $n \to \infty$, the set P has square-dispersion of order $O\big((\ln(n)/n)^{1/d}\big)$ (Deheuvels, 1983). Therefore, randomly-sampled points have asymptotically worse dispersion than center grids.



Numerical Sampling: Halton Sequences

Deterministic sampling sequences Halton sequences (Halton, 1960) are an elegant way of sampling an interval

- with good uniformity : better than a pseudorandom sequence, though not as good as the optimal center grid)
- with the incremental property (which the center grid does not possess).

Each scalar Halton sequence is generated by a prime number

$$\frac{1}{2},\quad \frac{1}{4},\frac{3}{4},\quad \frac{1}{8},\frac{5}{8},\frac{3}{8},\frac{7}{8},\quad \frac{1}{16},\frac{9}{16},\quad \dots.$$

$$\frac{1}{3},\frac{2}{3},\quad \frac{1}{9},\frac{4}{9},\frac{7}{9},\frac{2}{9},\frac{5}{9},\frac{8}{9},\quad \frac{1}{27},\quad \dots.$$

Halton sequence algorithm

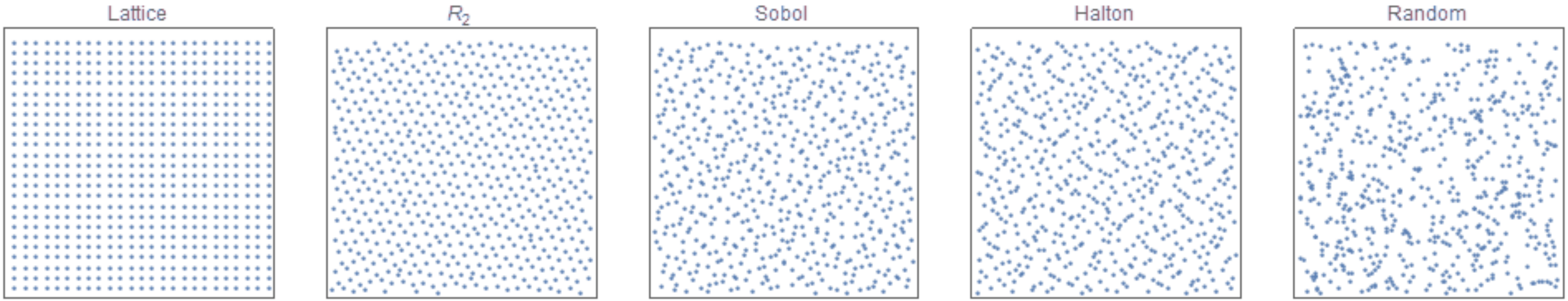
Input: length of the sequence $N \in \mathbb{N}$ and prime number $p \in \mathbb{N}$
Output: an array S with the first N samples of the Halton sequence generated by p

```
initialize: S to be an array of N zeros (i.e., S(i) := 0 for each i from 1 to N)
for each i from 1 to N :
    initialize: i_tmp := i, and f := 1/p
    while i_tmp > 0 :
        compute the quotient q and the remainder r of the division i_tmp/p
        S(i) := S(i) + f · r
        i_tmp := q
        f := f/p
return S
```

- square-dispersion of a Halton sequence of n samples is $f(d)/\sqrt[2]{n}$, where $f(d)$ is a constant for each dimension d.
- the Halton sequence achieves a dispersion similar to that of uniform grids, but has the advantage of allowing for incremental increases in the number of samples.

Numerical Sampling: Comparison

Sampling property	Uniform grids	Random sampling	Halton sequences
dispersion	$O\left(\frac{1}{\sqrt[d]{n}}\right)$	$O\left(\frac{\ln^{1/d}(n)}{\sqrt[d]{n}}\right)$	$O\left(\frac{f(d)}{\sqrt[d]{n}}\right)$
incremental	no	yes	yes
lattice	yes	no	yes (more complex)



http://extremelearning.com.au/a-simple-method-to-construct-isotropic-quasirandom-blue-noise-point-sequences/

Numerical Collision Detection Methods

Problem 4.2 Given two bodies B1 and B2, determine if they collide. (In equivalent set-theoretic words, determine if the intersection between two sets is non-empty.)

The distance between two sets A and B, in a norm ||.||, is defined as

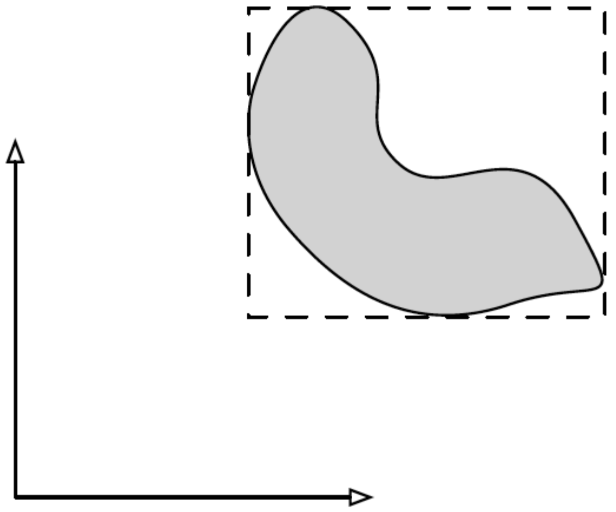
$$dist(A,B) = \inf\{\|p-q\| \mid p \in A, q \in B\}.$$

The two sets A and B do not intersect if $dist(A,B) > 0$.

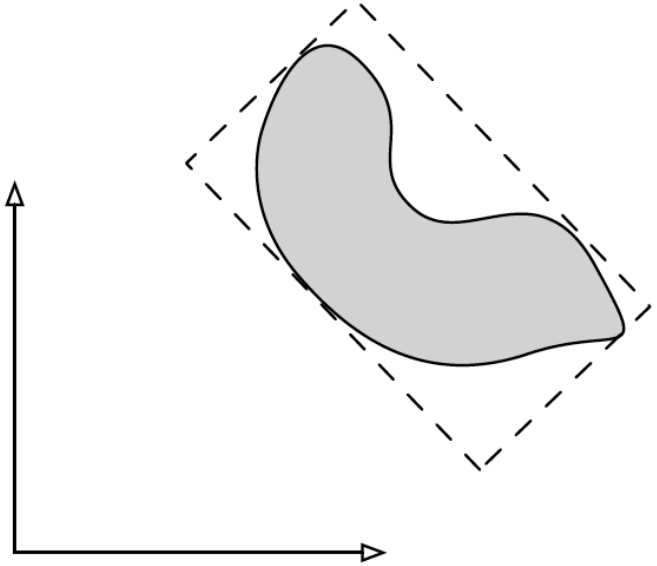
It is undesirable to check collision by computing the pairwise distance between any two points. Therefore, it is convenient to devise careful algorithms for collision detection.

Numerical Collision Detection Methods: Approximate Methods

Use well-defined geometric bounding shapes to over-approximate sets in collision detection problems

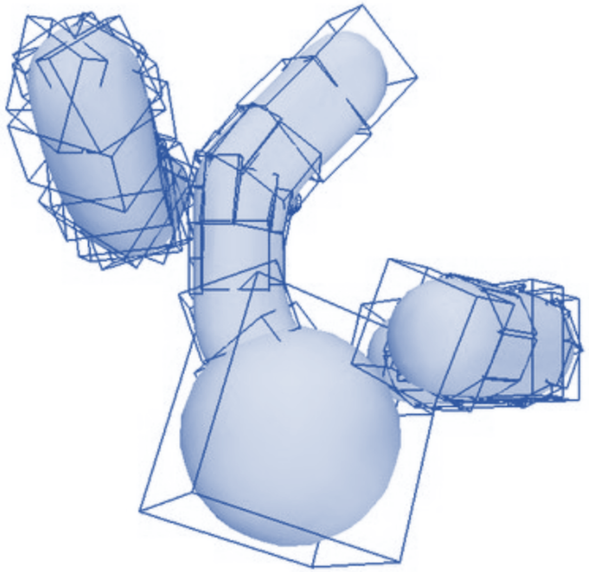


Axis-aligned Bounding Box (AABB)



Oriented Bounding Box (OBB)

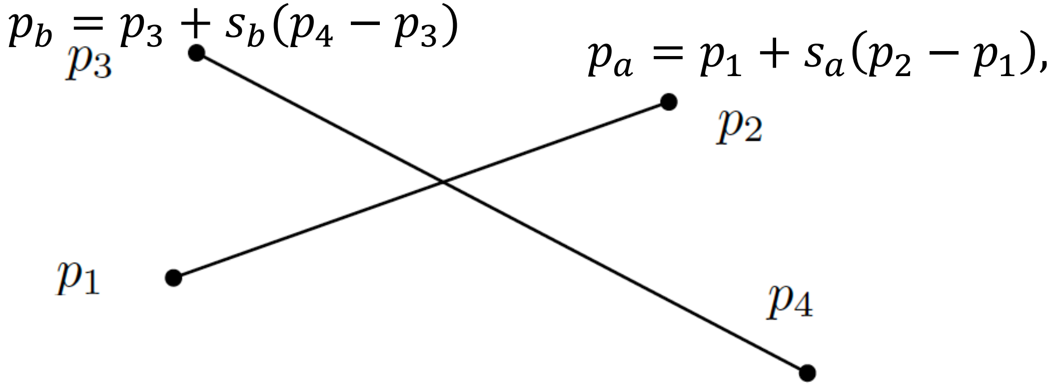
(i) bounding spheres, rather than
(ii) Axis-Aligned Bounding Boxes (AABB), rather than
(iii) Oriented Bounding Boxes, rather than
(iv) convex polygons, rather than
(v) non-convex polygons, rather than
(vi) arbitrary shapes.



Convex decomposition is used to decompose arbitrary, triangular meshes into approximate convex pieces and their corresponding Oriented Bounding Boxes or OBBs (shown in blue) for efficient collision detection detection and distance queries.
Courtesy of: 10.1109/BIOROB.2010.5625965

Basic primitive #1: do two segments intersect?

Problem 4.3 Given two segments, determine if they intersect.



Testing for the intersection of two segments

These two equations can be solved and, for example, one obtains

$$s_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} =: \frac{\texttt{num}}{\texttt{den}}.$$

One can show that

(i) if $\texttt{num} = \texttt{den} = 0$, then the two lines are coincident,

(ii) if $\texttt{num} \neq 0$ and $\texttt{den} = 0$, then the two lines are parallel and distinct, and

(iii) if $\texttt{den} \neq 0$, then the two lines are not parallel and therefore intersect at a single point.

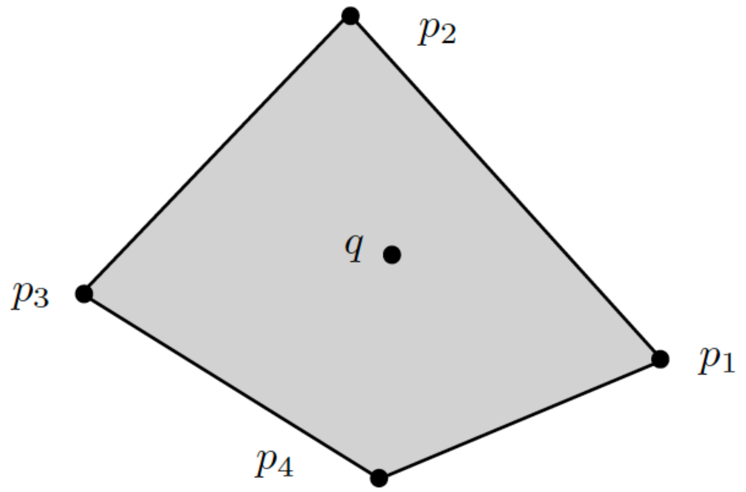
Does the intersection point actually belongs to the segment?

If $s_a \in [0,1]$ and $s_b \in [0,1]$, the two segments collide!

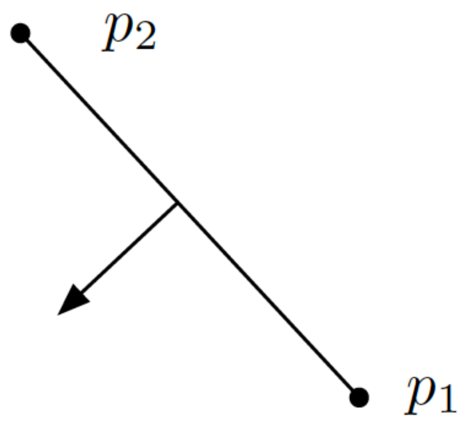
Basic primitive #2: is a point in a convex polygon?

Problem 4.4 Given a convex polygon and a point, determine if the point is inside the polygon.

The polygon is defined by a counter-clockwise sequence of vertices, $p_1, \ldots, p_4$ in Figure 4.19. For each side of the polygon, we define the *interior normal* as in Figure 4.20 for the side $\overline{p_1p_2}$. Notice that the interior normal of $\overline{p_ip_{i+1}}$ is obtained by rotating the vector $(p_{i+1} - p_i)$ by the angle $\pi/2$. Letting $p_i = (x_i, y_i)$ and $p_{i+1} = (x_{i+1}, y_{i+1})$, the interior normal is $\big(-(y_{i+1} - y_i), (x_{i+1} - x_i)\big)$.



Testing if a point lies in a polygon



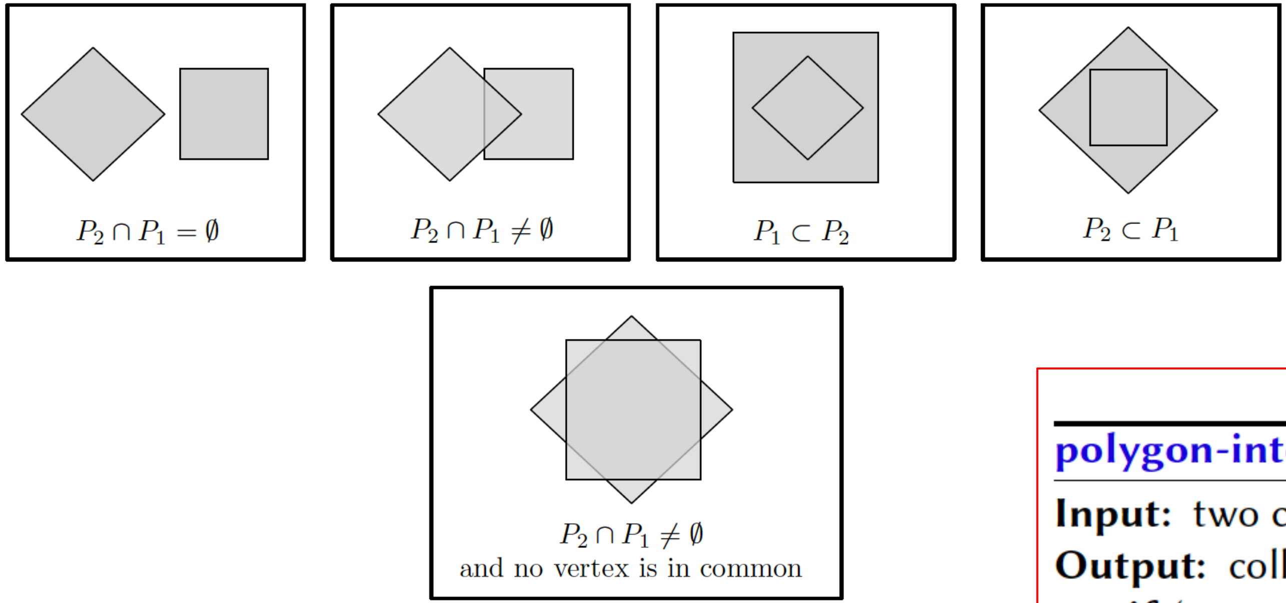
Interior normal to a side of the polygon

Given a polygon with vertices $p_1, \cdots, p_n$, labeled counter clockwise and $p_{n+1} = p1$, the point q is in the polygon (possibly on the boundary), if and only if for all $i \in \{1, \cdots, n\}$, the dot product between the interior normal to the side $\overline{p_ip_{i+1}}$ and the segment $\overline{p_iq}$ is positive or zero.

Numerical Collision Detection Methods: Basic primitives

Basic primitive #3: do two convex polygons intersect?

Problem 4.5 Given two convex polygons, determine if they intersect.



five possible cases that one must consider

polygon-intersection algorithm

Input: two convex polygons P_1 and P_2
Output: collision or no collision

```
if (any vertex of P_1 belongs to P_2) OR (any vertex of P_2 belongs to P_1) :
        return collision
if any edge of P_1 intersects any edge of P_2 :
        return collision
return no collision
```

Numerical Collision Detection Methods: Basic primitives

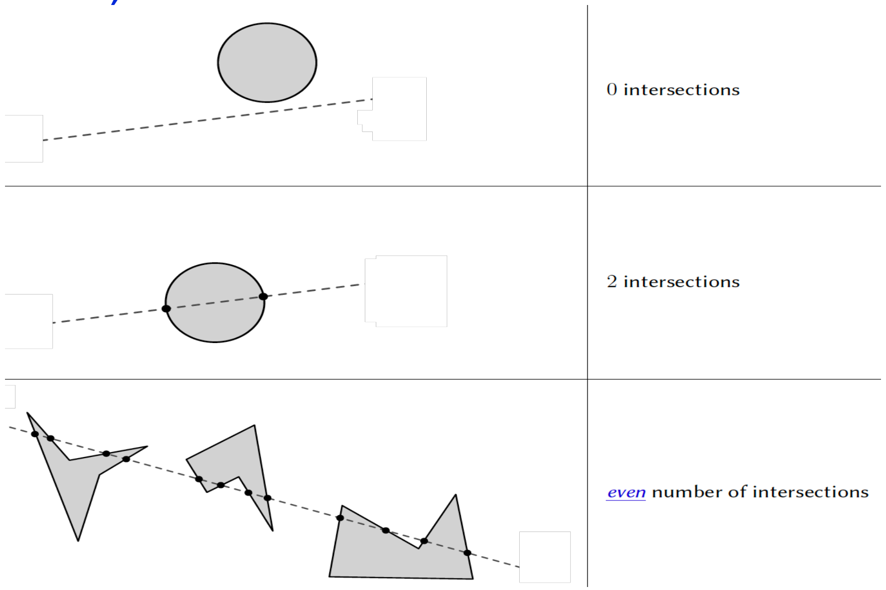
Basic primitive #3: do two nonconvex polygons intersect?

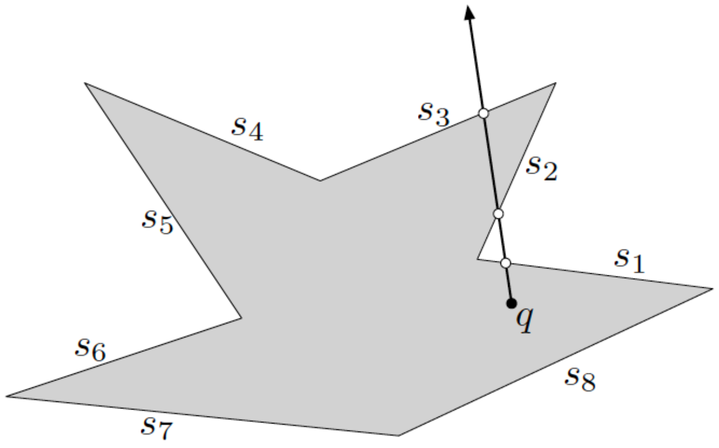
Problem 4.6 Given a non-convex polygon and a point, determine if the point is inside the polygon.

Problem 4.7 Given a line segment s and a ray R, determine if they intersect.

Problem 4.8 Given two non-convex polygons, determine if they intersect.

By virtue of Jordan Theorem, the number of intersections between a line segment and a closed curve, where the endpoints of the segment lie on the outside of the curve, is even





Shooting ray method

The key idea is ☝, rest left to you to figure out ☺

Collision Detection Methods: Final Note

(i) collision detection algorithms for simple objects are easy to perform,

(ii) for complex objects, e.g., arbitrary shapes, a reasonable approach is to use hierarchical approximations and decompositions, described as follows:

(a) approximate the complex shape by a simple enclosing shape, e.g., a sphere, an AABB, or an OBB,

(b) if no collision occurs between the two simple enclosing shapes, then return a “no collision” result,

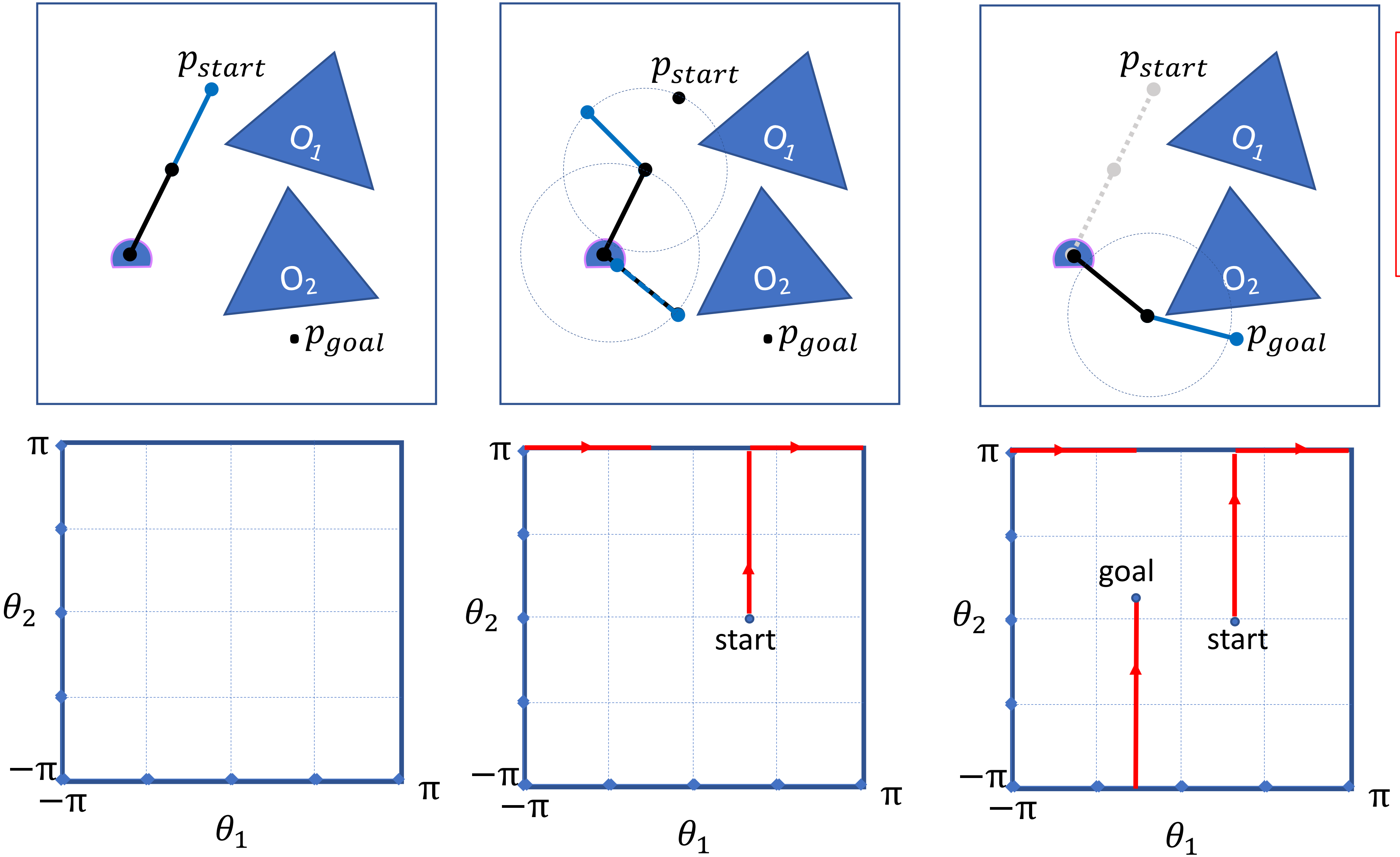
(c) if a collision is detected between two simple enclosing shapes, then approximate the bodies less conservatively and more accurately, e.g., by decomposing them into the union of multiple simple shapes. One can then check collision between these more accurate decompositions;

(iii) to detect collisions between moving objects, discretize time and perform a collision detection test for each time step.

References:

- F. Bullo and S. L. Smith. Lecture notes on robotic planning and kinematics
- H. Choset, K. Lynch, S. Hutchinson, G. Kantor, et al. Principles of Robot Motion, Theory, Algorithms, and Implementations.

Motion planning for robots with finite shape and size



- A workspace $W \subset R^2$;
- Some obstacles $O_1, O_2, \cdots, O_n$;
- $W_{free} = W \backslash (O_1 \cup O_2 \cup \cdots \cup O_n)$

- A configuration space;
- …..
- $Q_{free} = \cdots$

Motion planning for rigid body robots: given a motion planning problem in the workspace "move from point $p_{start} \in W$ to point $P_{goal} \in W$," we need to translate this specification into the configuration space, i.e., move from a configuration $q_{start} \in Q$ to a configuration $q_{goal} \in Q$.