



JRL302: Robotics Technologies

Lecture 1: Motion Planning Module

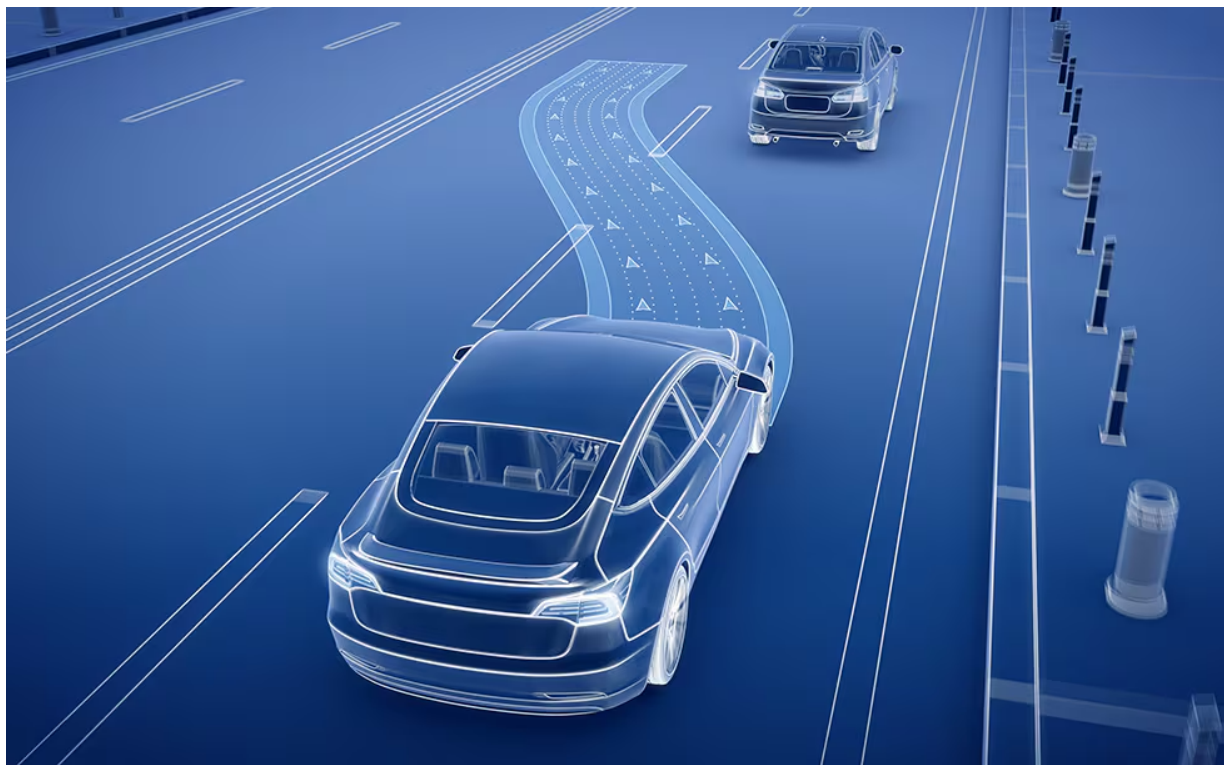
Mayank Sewlia
sewlia@iitd.ac.in

Motion Planning Module Objectives

- Overview of motion planning
- Basic concepts for motion planning problems
- Common graph-based and sampling-based algorithms
- Modern motion planning with kinodynamic constraints
- Current research outlook

This Class

- What is motion planning?
- Basic motion planning problem
- Types and properties of motion planning problems
- Representation of obstacles
- Some demos



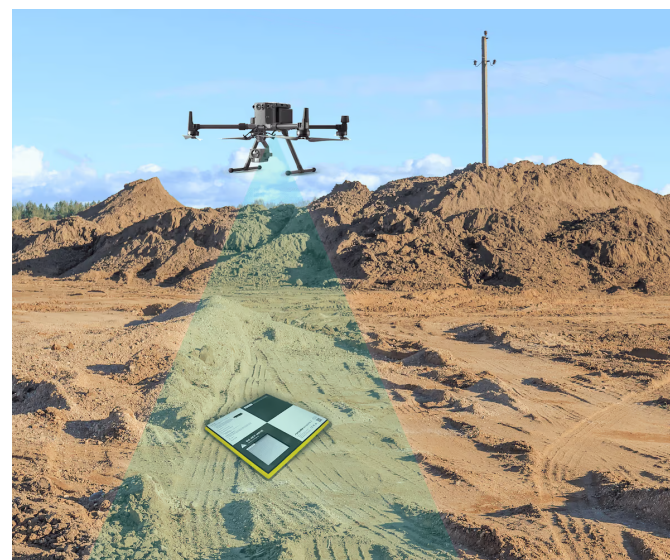
source: TDK



source: SpaceX



source: Neura Robotics

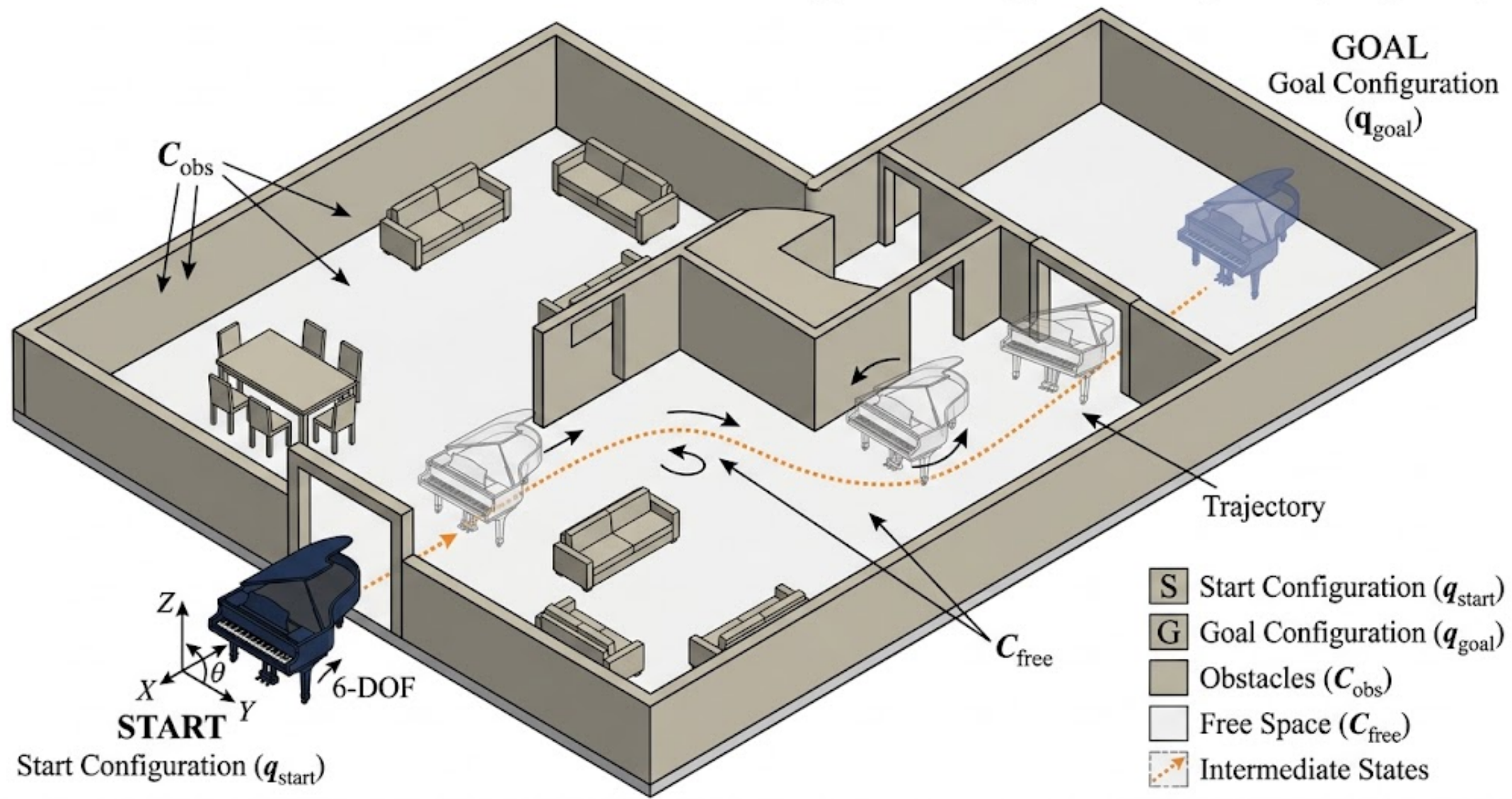


source: Propeller Aero



source: ORI

Piano Mover's Problem



This Class

- What is motion planning?
- Basic motion planning problem
- Types and properties of motion planning problems
- Representation of obstacles
- Some demos

Basic Motion Planning Problem

1. Configuration

$$q = (x, y, \theta) \in SE(2)$$

2. Kinematics

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

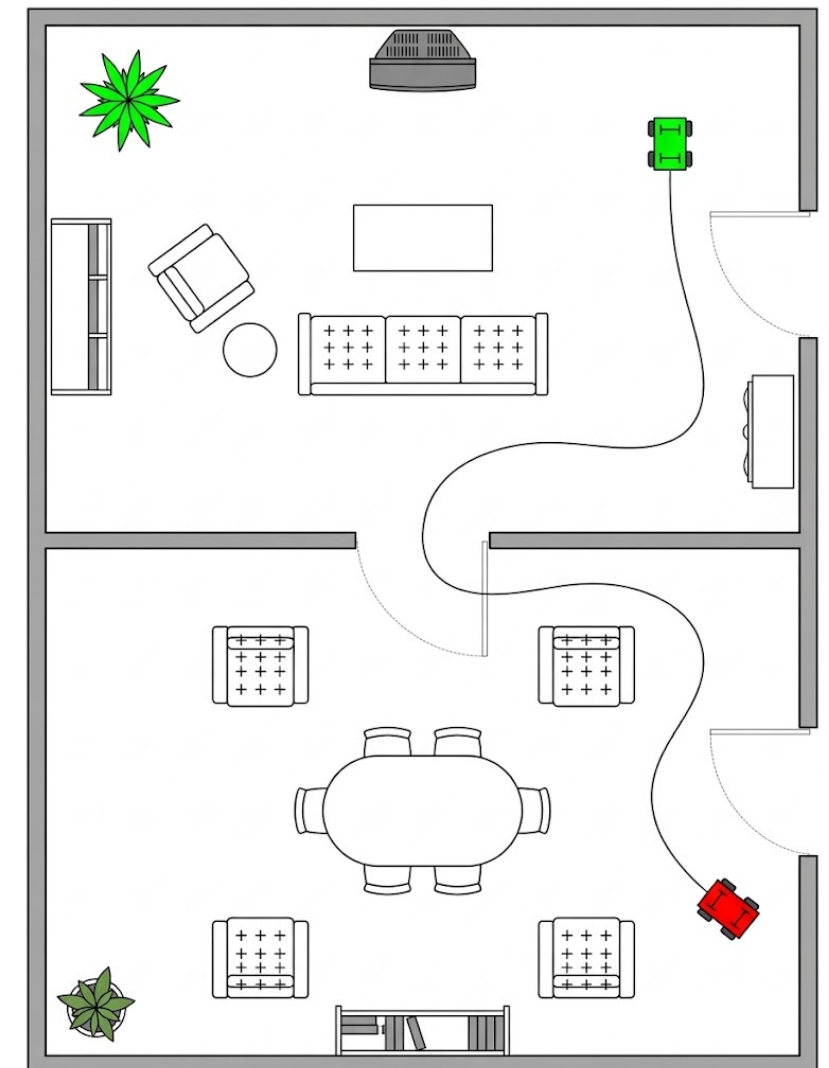
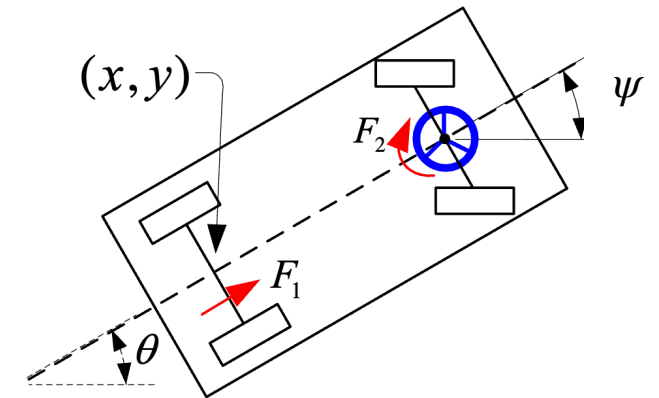
$$\dot{\theta} = \frac{v}{L} \tan \psi$$

3. Dynamics

$$z = (x, y, \theta, v, \psi)$$

$$u = (F, \omega_\psi)$$

$$\dot{z} = f(z, u)$$



Basic Motion Planning Problem

1. Configuration

$$q = (x, y, \theta) \in SE(2)$$

Given,

2. Kinematics

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \frac{v}{L} \tan \psi$$

- Robotic system
- Work environment
- Initial state
- Goal state

3. Dynamics

$$z = (x, y, \theta, v, \psi)$$

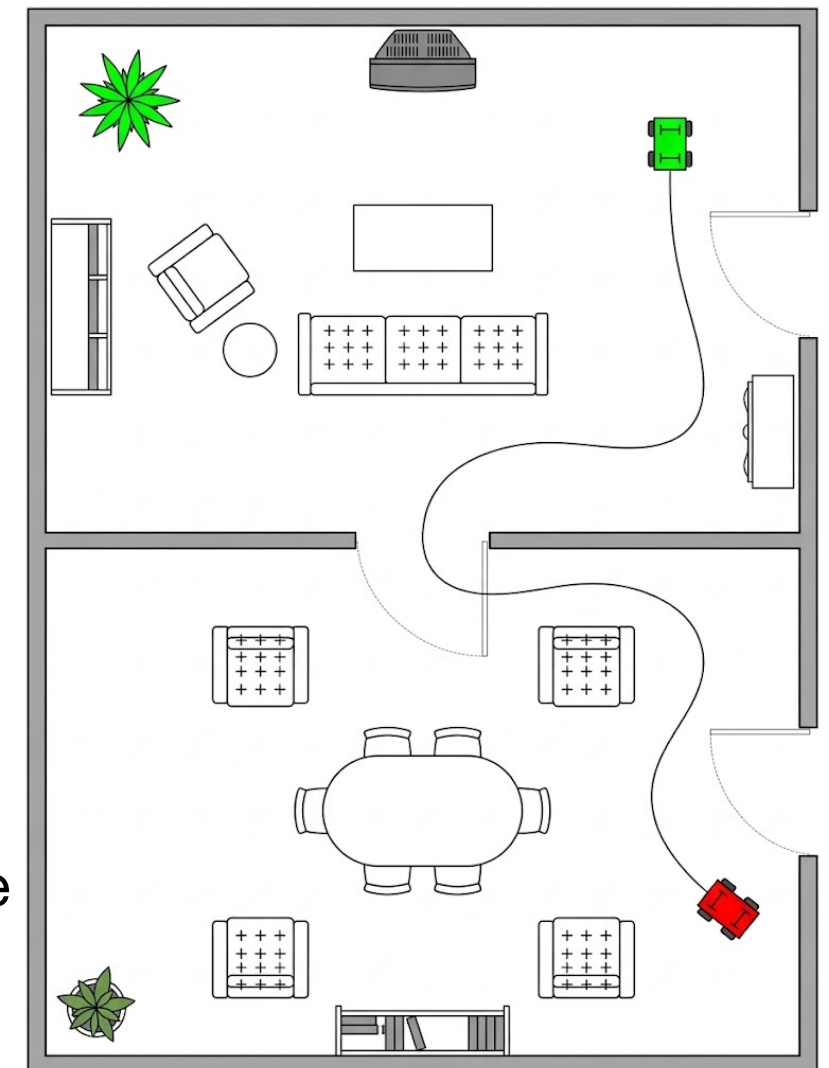
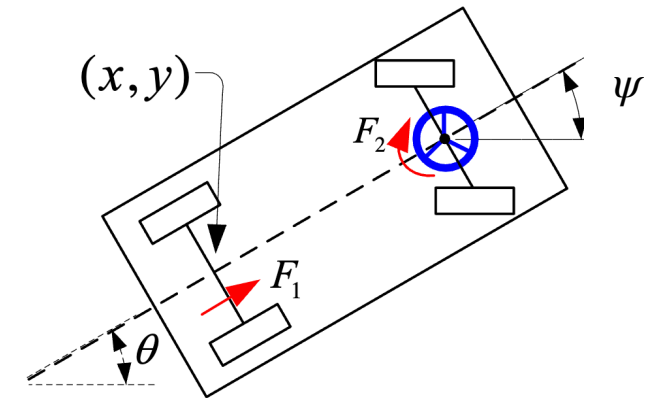
$$u = (F, \omega_\psi)$$

$$\dot{z} = f(z, u)$$

Find,

a control, $u : [0, T] \rightarrow \mathcal{U}$

and trajectory, $q : [0, T] \rightarrow \mathcal{C}_{\text{free}}$



Basic Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

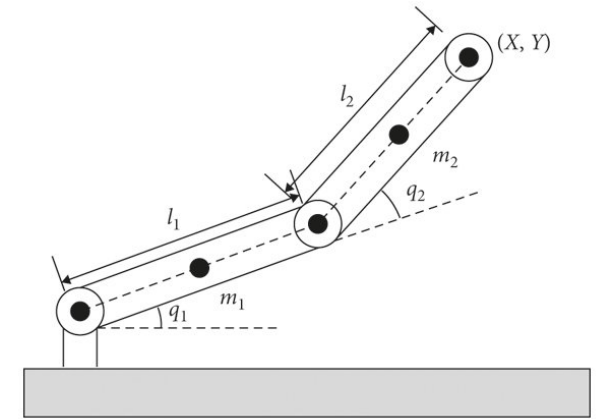
$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$



Basic Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1 \quad \text{Given,}$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

- Robotic system
- Work environment
- Initial state
- Goal state

3. Dynamics

$$z = (q, \dot{q})$$

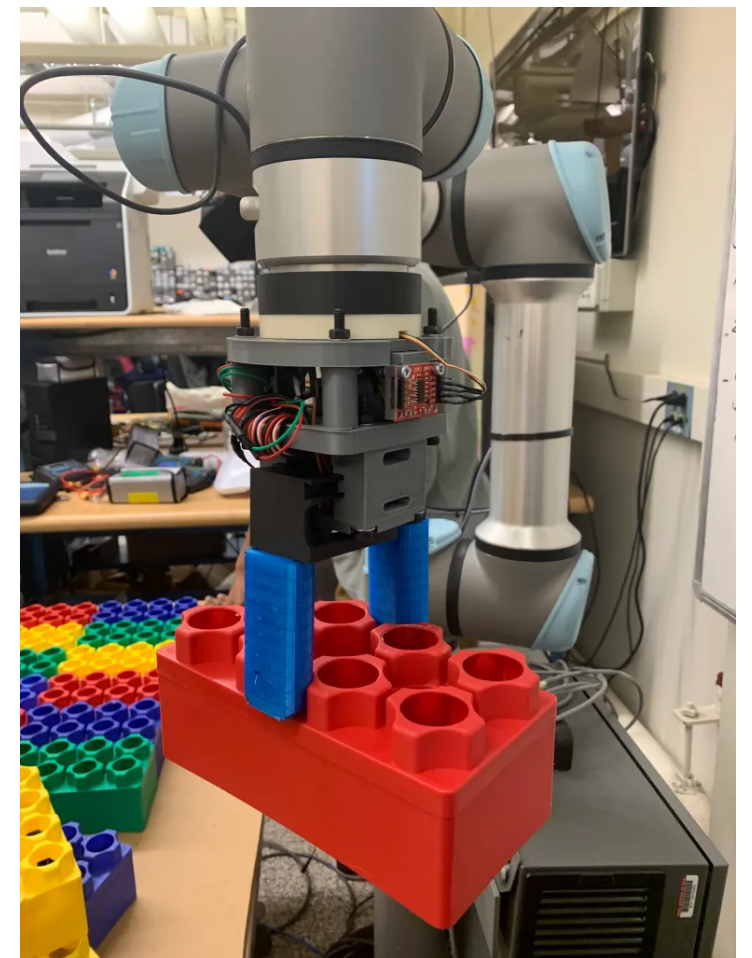
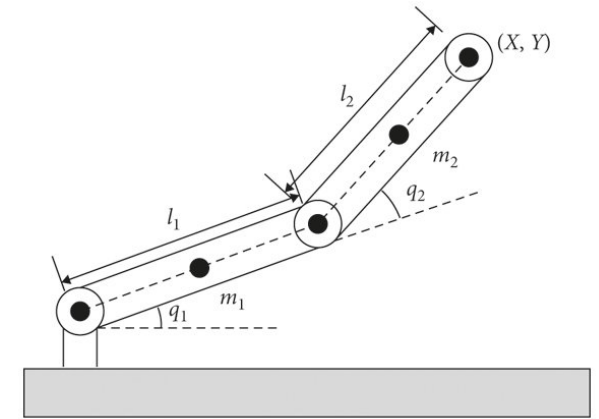
$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Find,

a control, $u : [0, T] \rightarrow \mathcal{U}$

and trajectory, $q : [0, T] \rightarrow \mathcal{C}_{\text{free}}$



This Class

- What is motion planning?
- Basic motion planning problem
- Types and properties of motion planning problems
- Representation of obstacles
- Some demos

Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1 \quad \text{Given,}$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

- Robotic system
- Work environment
- Initial state
- Goal state

3. Dynamics

$$z = (q, \dot{q})$$

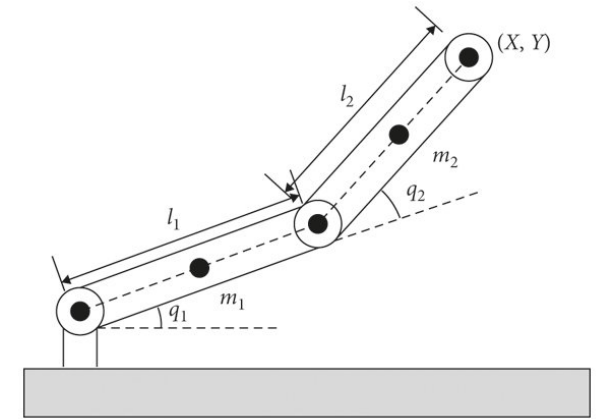
$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Find,

a control, $u : [0, T] \rightarrow \mathcal{U}$

and trajectory, $q : [0, T] \rightarrow \mathcal{C}_{\text{free}}$



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

Given,

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

- Robotic system
- Work environment
- Initial state
- Goal state

3. Dynamics

$$z = (q, \dot{q})$$

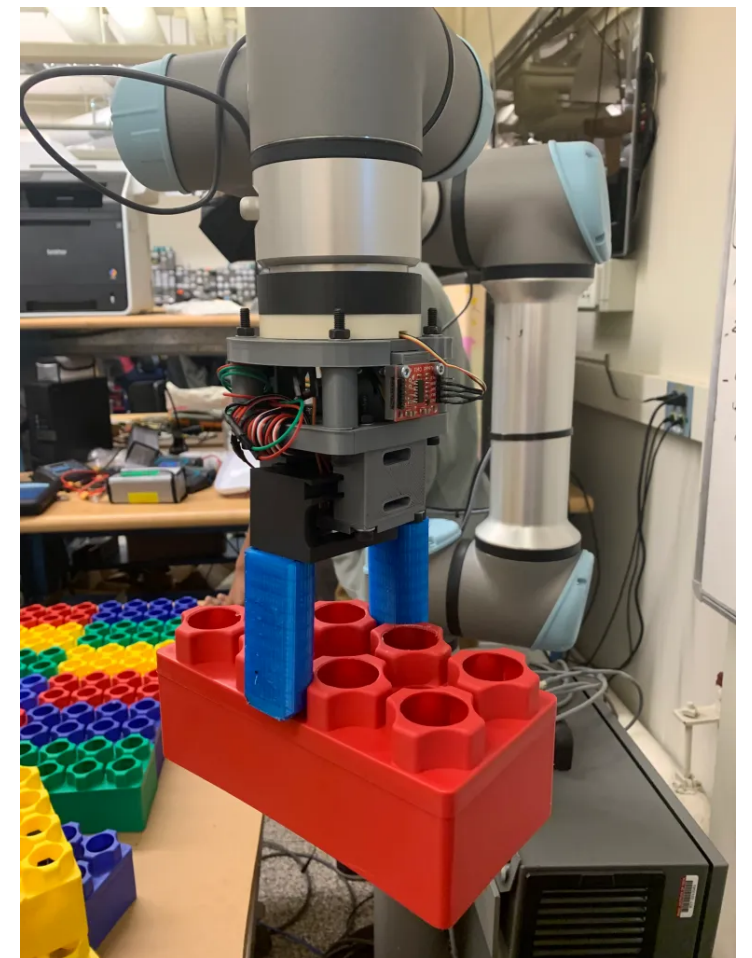
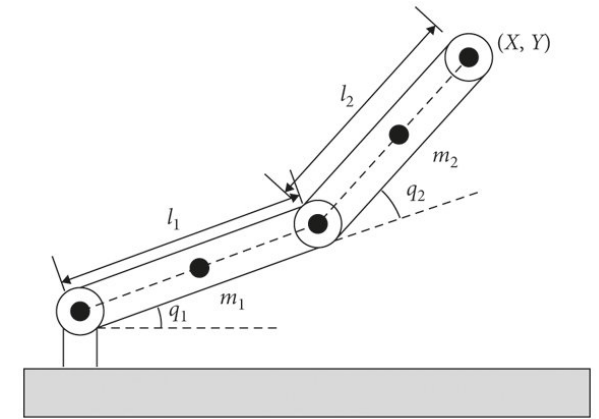
$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Find,

a control, $u : [0, T] \rightarrow \mathcal{U}$

and trajectory, $q : [0, T] \rightarrow \mathcal{C}_{\text{free}}$



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Path Planning Problem

Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

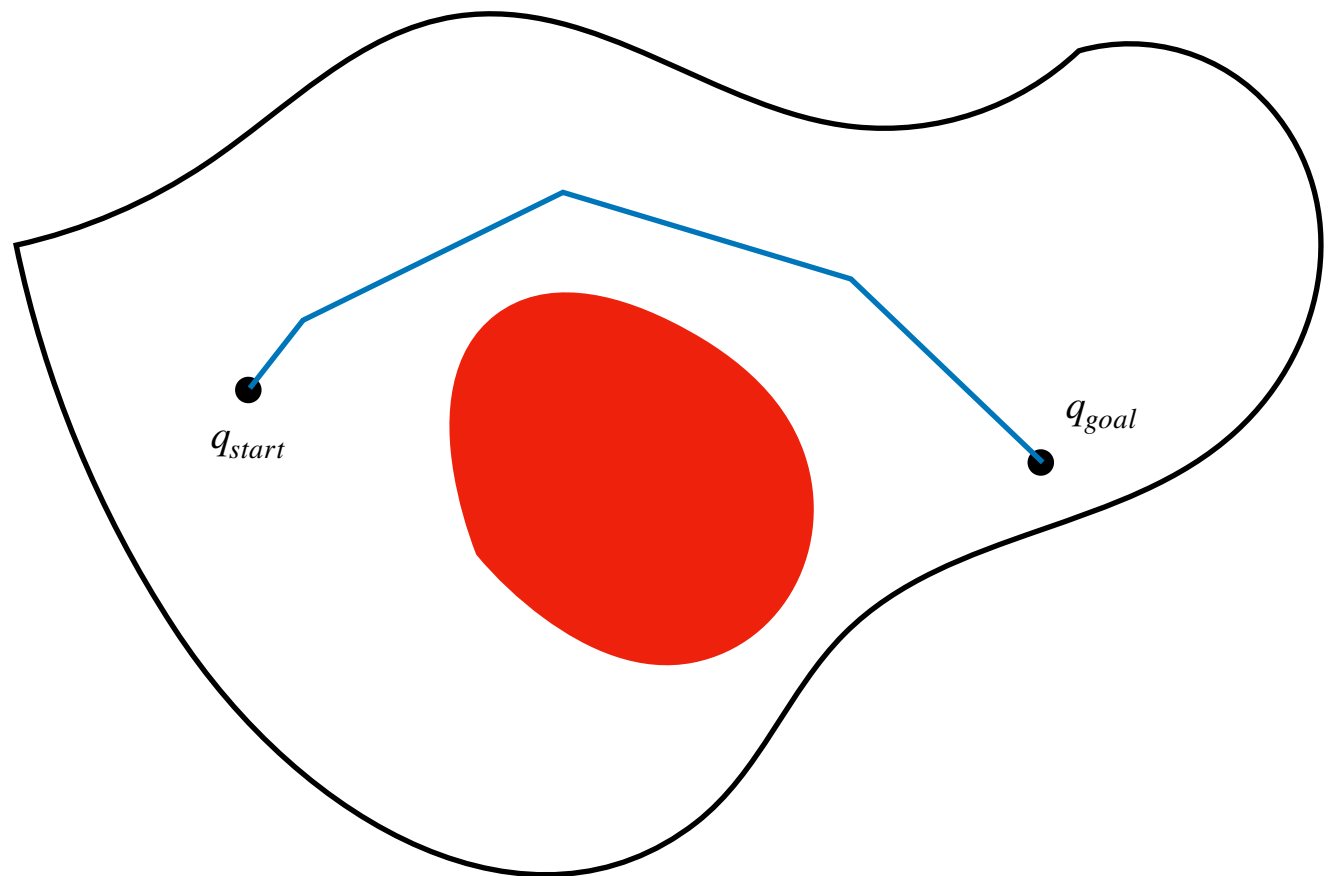
3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Path Planning Problem



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Path Planning Problem

But will slide sideways!



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

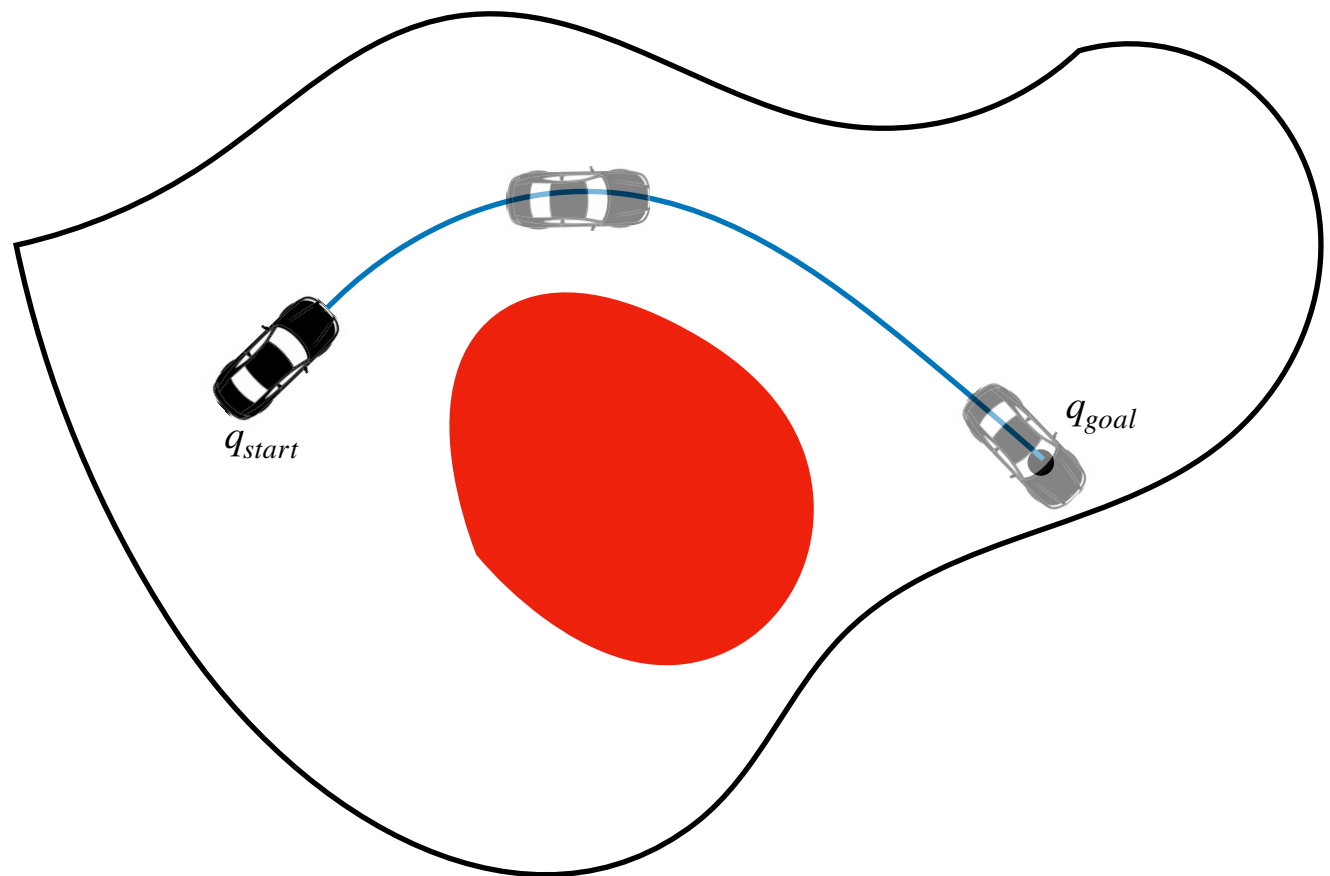
3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Kinematic/Nonholonomic Planning Problem



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Kinematic/Nonholonomic Planning Problem



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

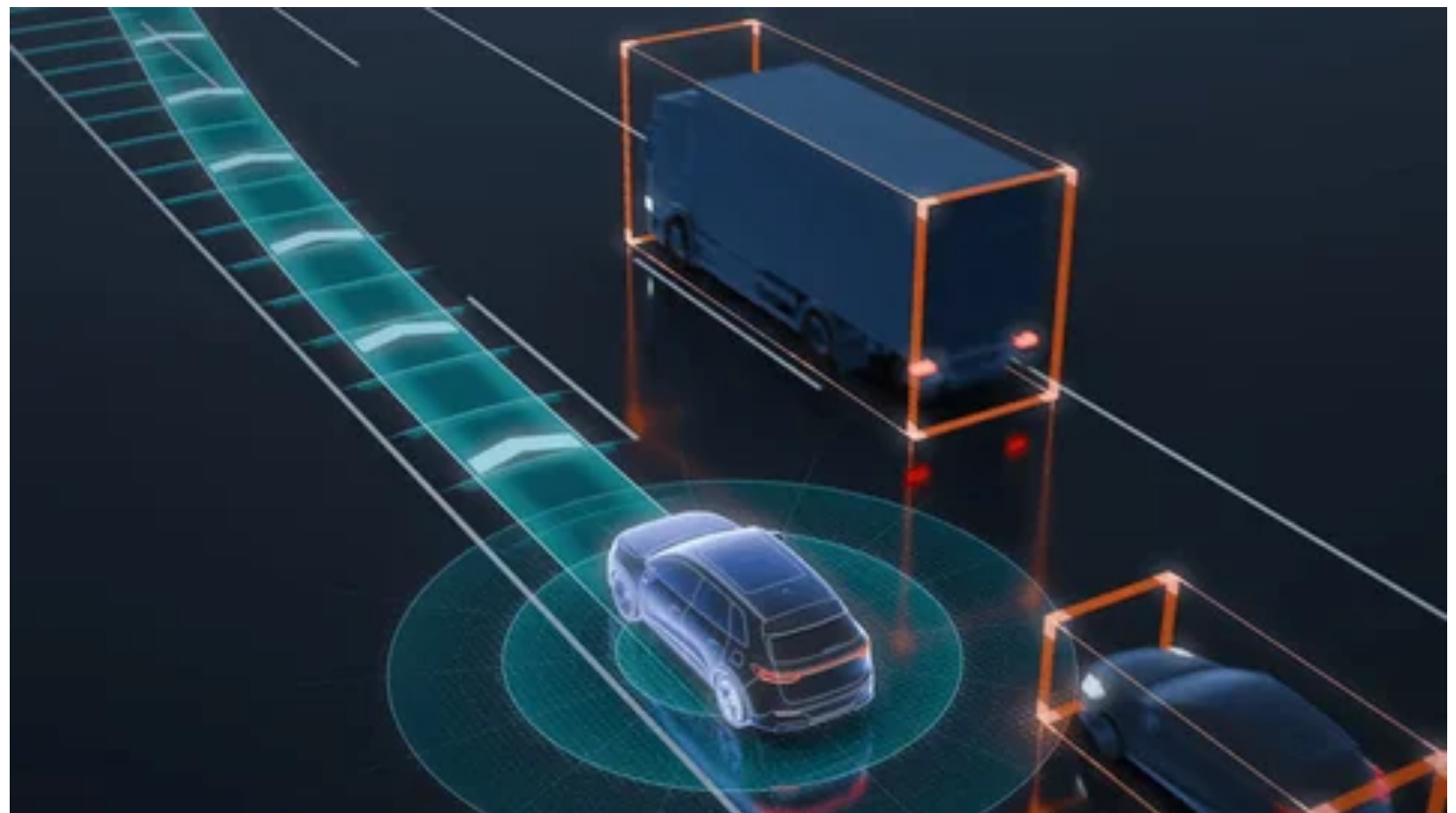
$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Kinematic/Nonholonomic Planning Problem

May not have enough thrust to overtake based on the plan!



Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

Kinodynamic Planning

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Types of Motion Planning Problem

1. Configuration

$$q = (q_1, q_2) \in \mathbb{S}^1 \times \mathbb{S}^1$$

2. Kinematics

$$X = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$Y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

$$\dot{p} = J(q)\dot{q}$$

3. Dynamics

$$z = (q, \dot{q})$$

$$u = \tau$$

$$\dot{z} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \end{bmatrix} = f(z, u)$$

Kinodynamic Planning

Most of the times intractable!

Types of Motion Planning Problem

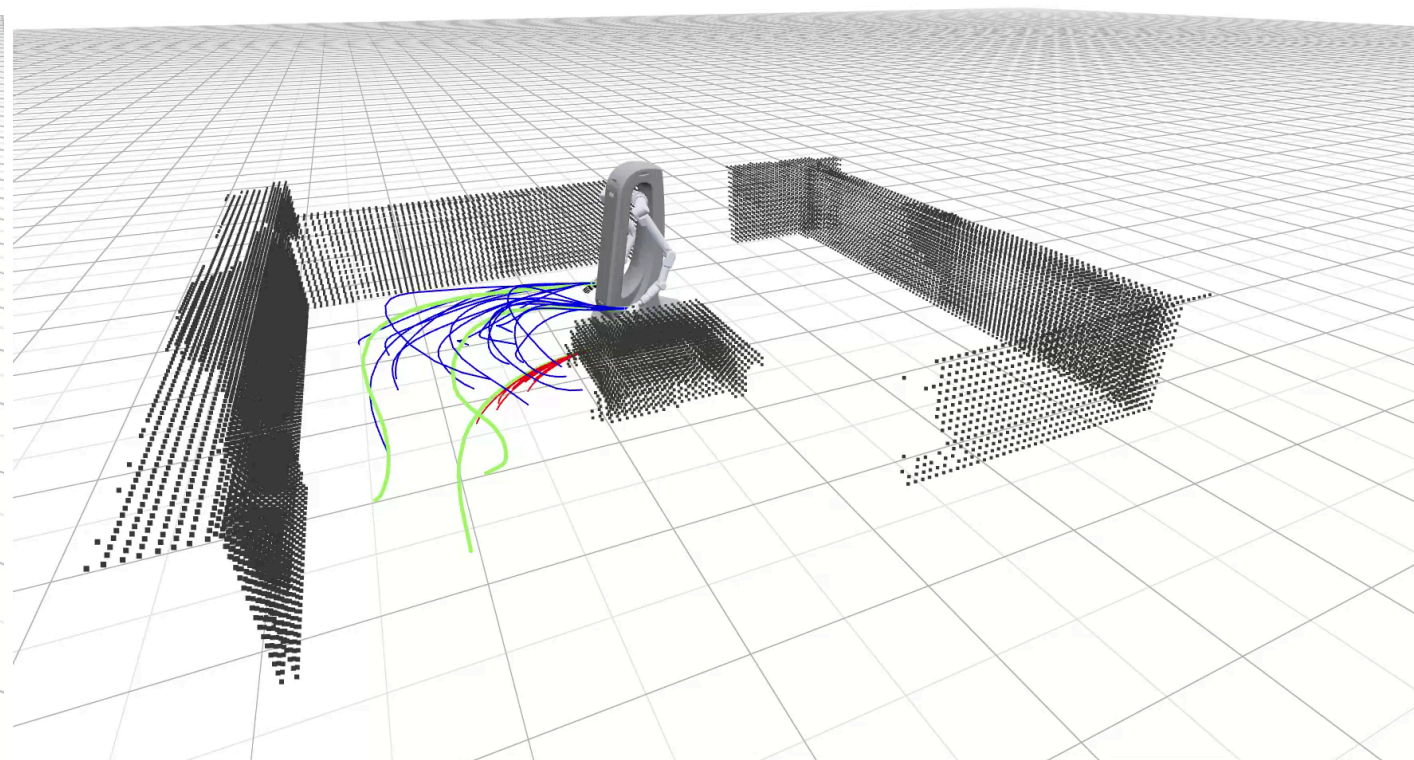
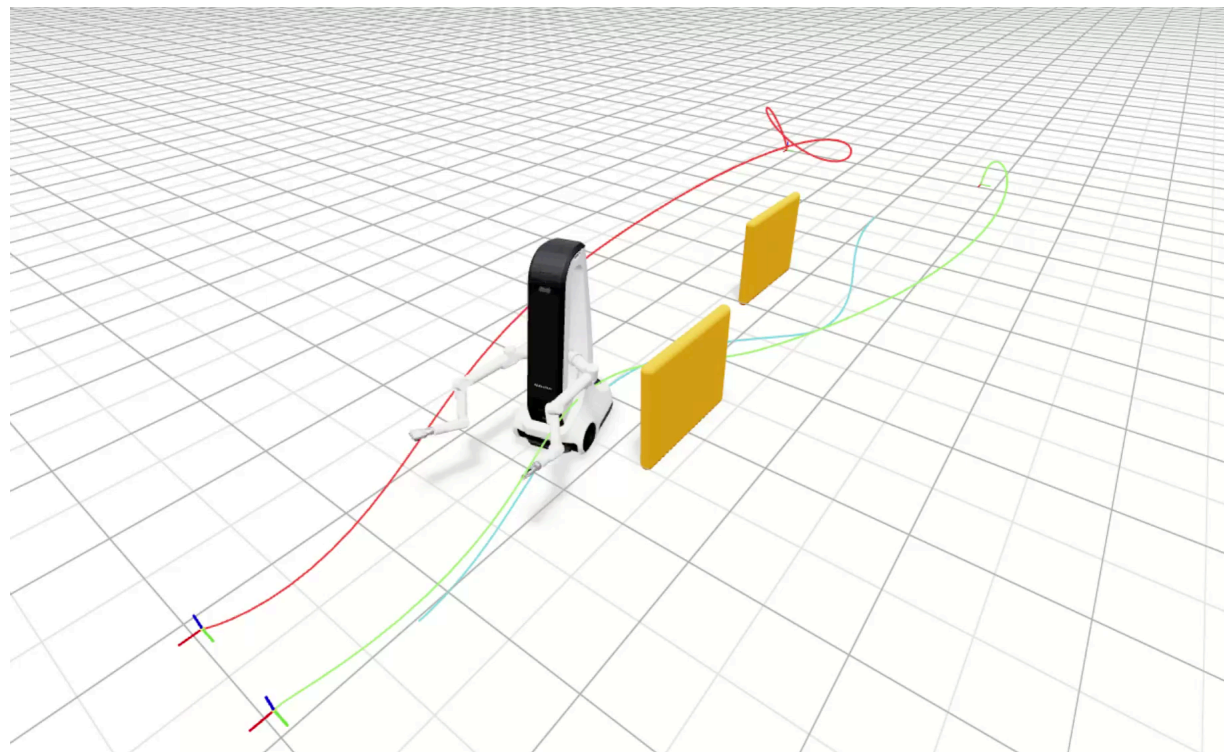
1. Path planning
2. Kinematic/nonholonomic planning
3. Kinodynamic planning

1. Offline planning
2. Online planning

Types of Motion Planning Problem

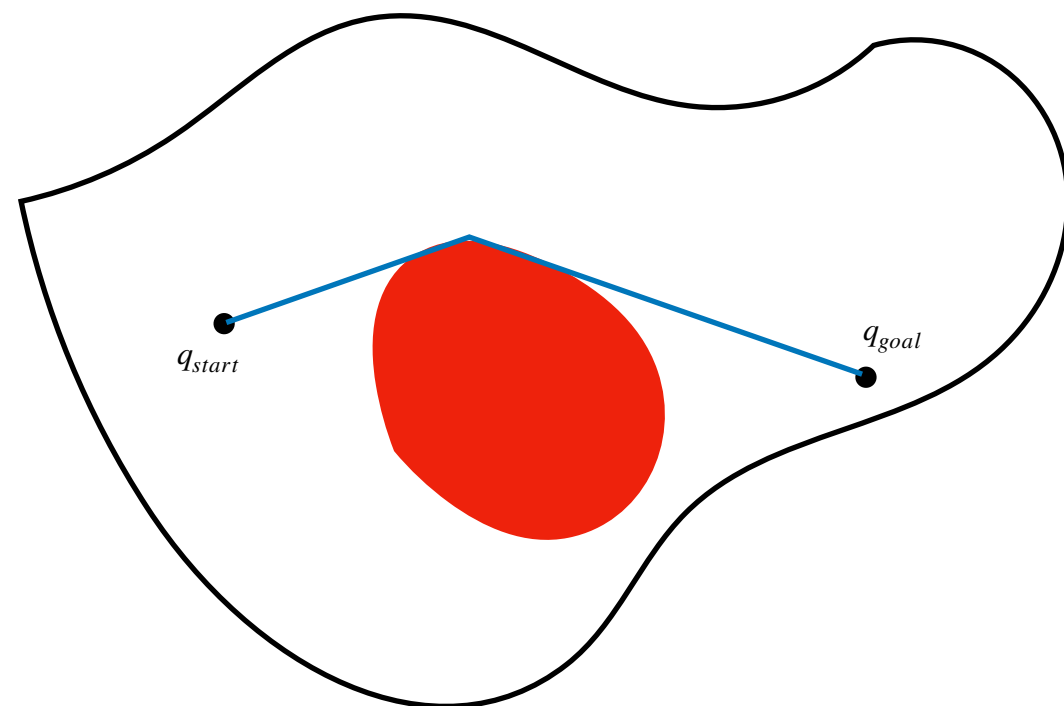
1. Path planning
2. Kinematic/nonholonomic planning
3. Kinodynamic planning

1. Offline planning
2. Online planning

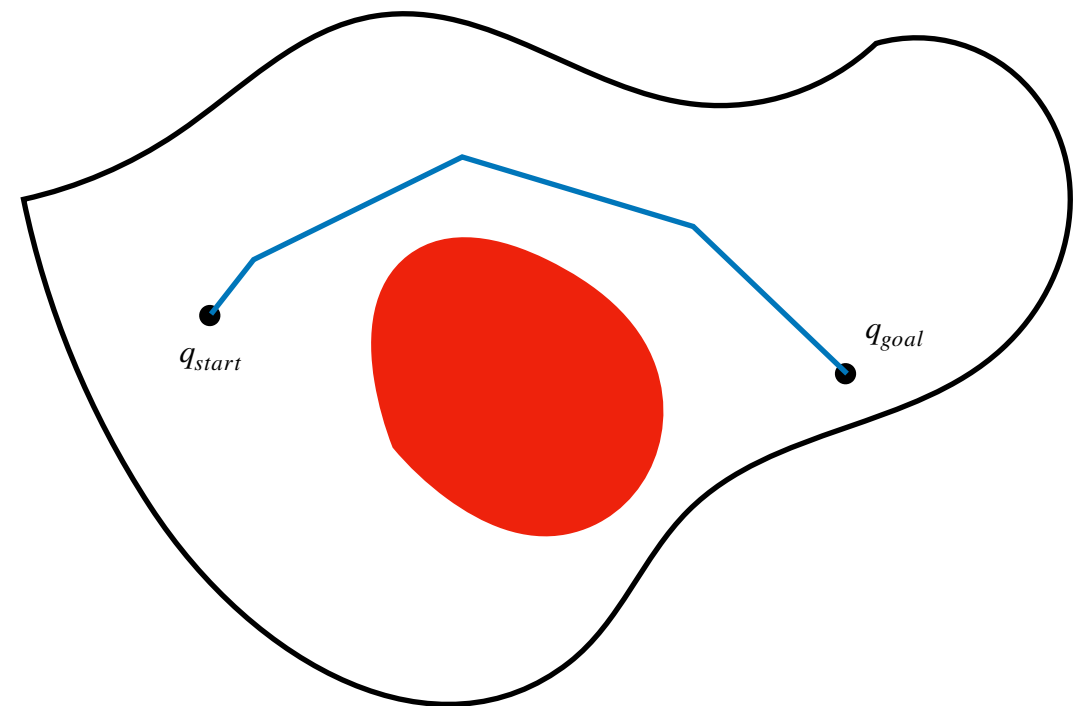


Types of Motion Planning Problem

1. Path planning
2. Kinematic/nonholonomic planning
3. Kinodynamic planning



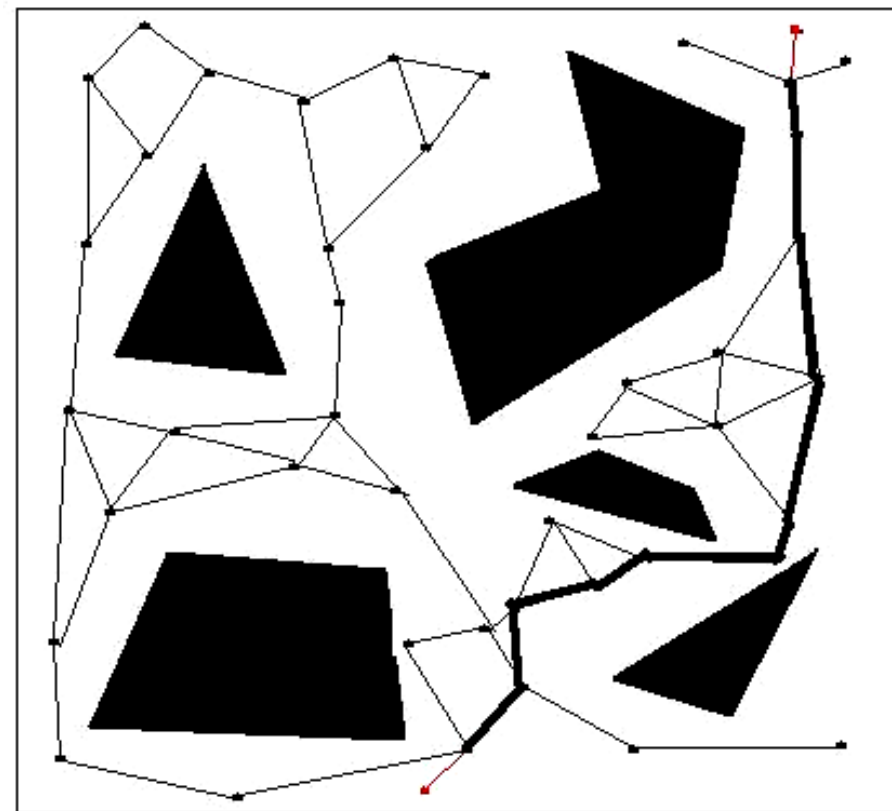
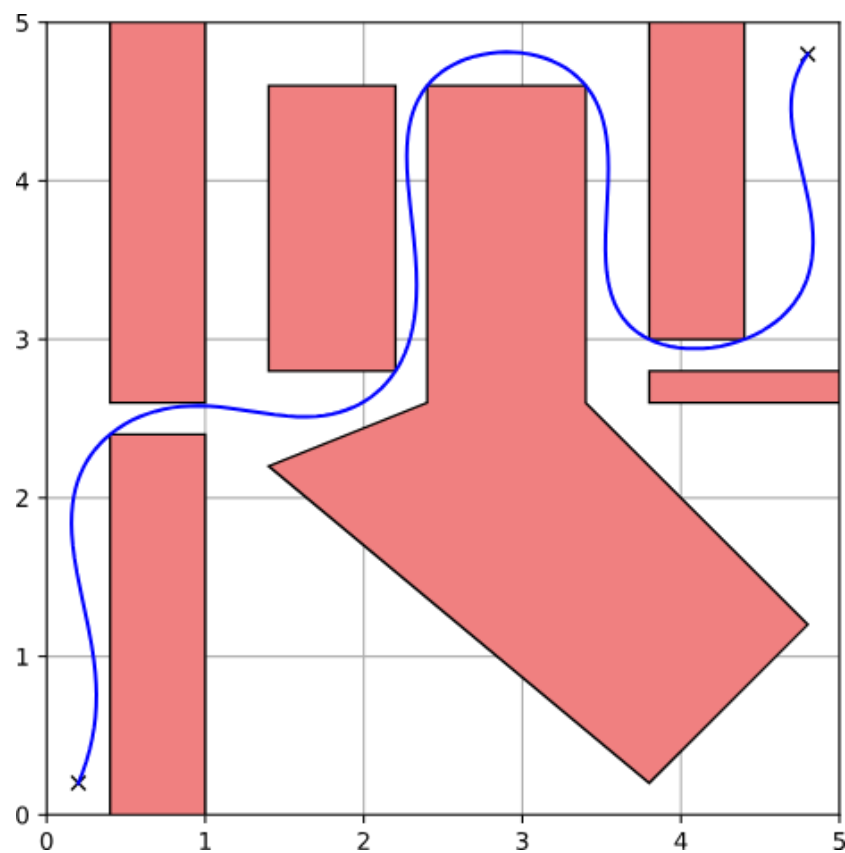
1. Offline planning
2. Online planning



1. Optimal planning
2. Satisfying planning

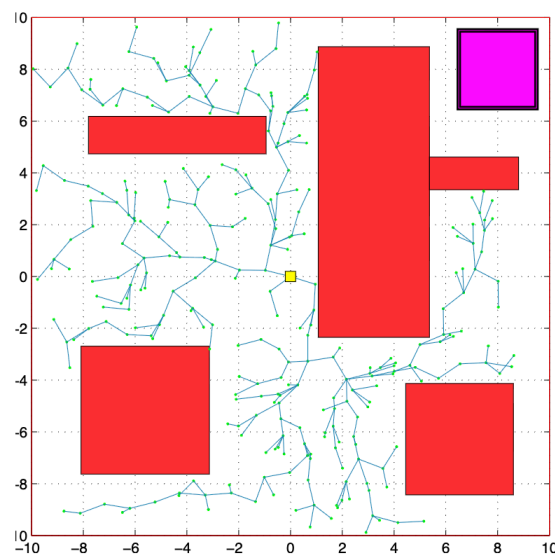
Properties of Motion Planning Problem

1. Single query planner
2. Multi-query planner

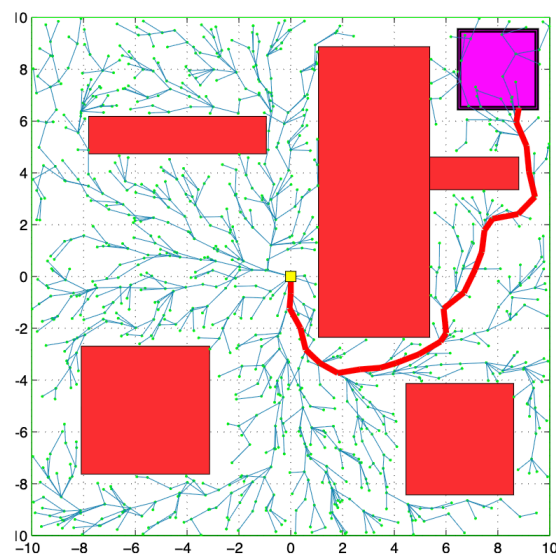


Properties of Motion Planning Problem

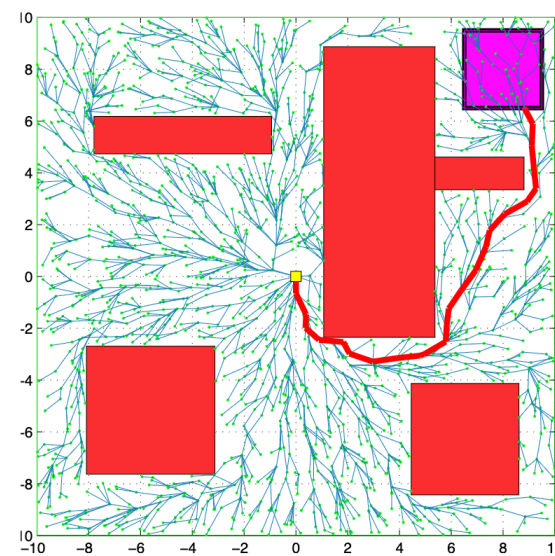
"Anytime planning"



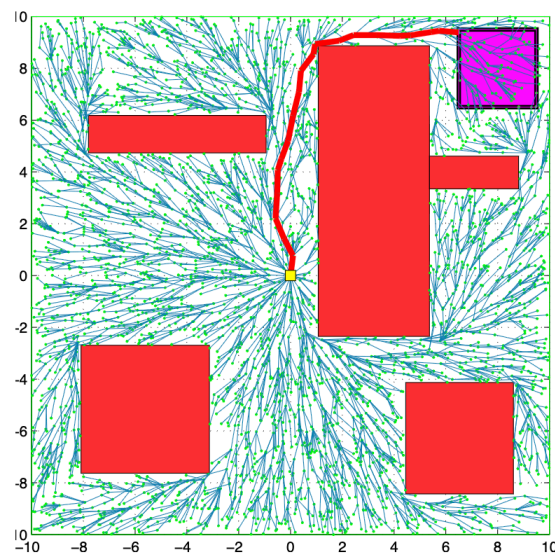
(a)



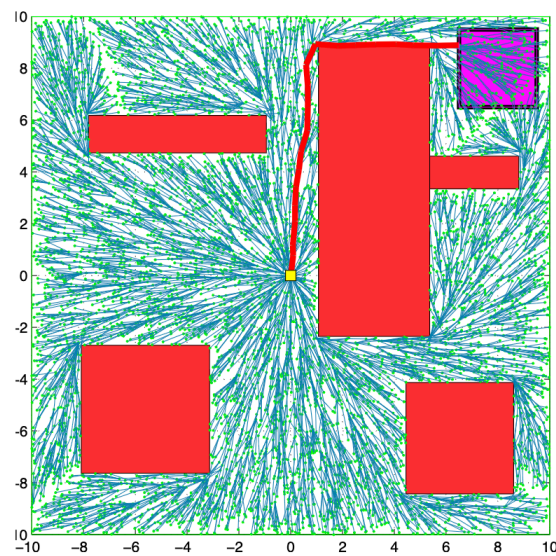
(b)



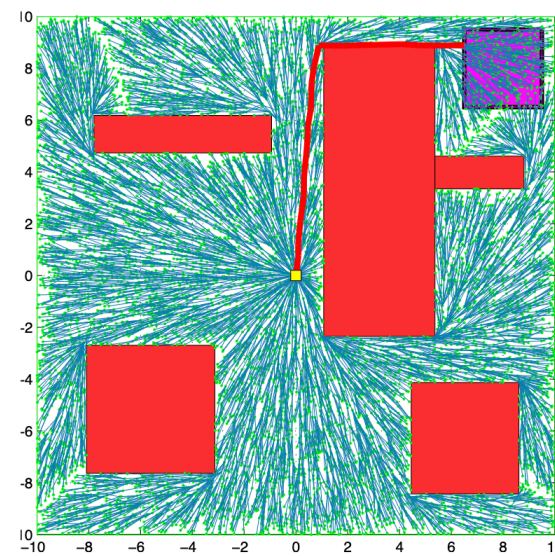
(c)



(d)

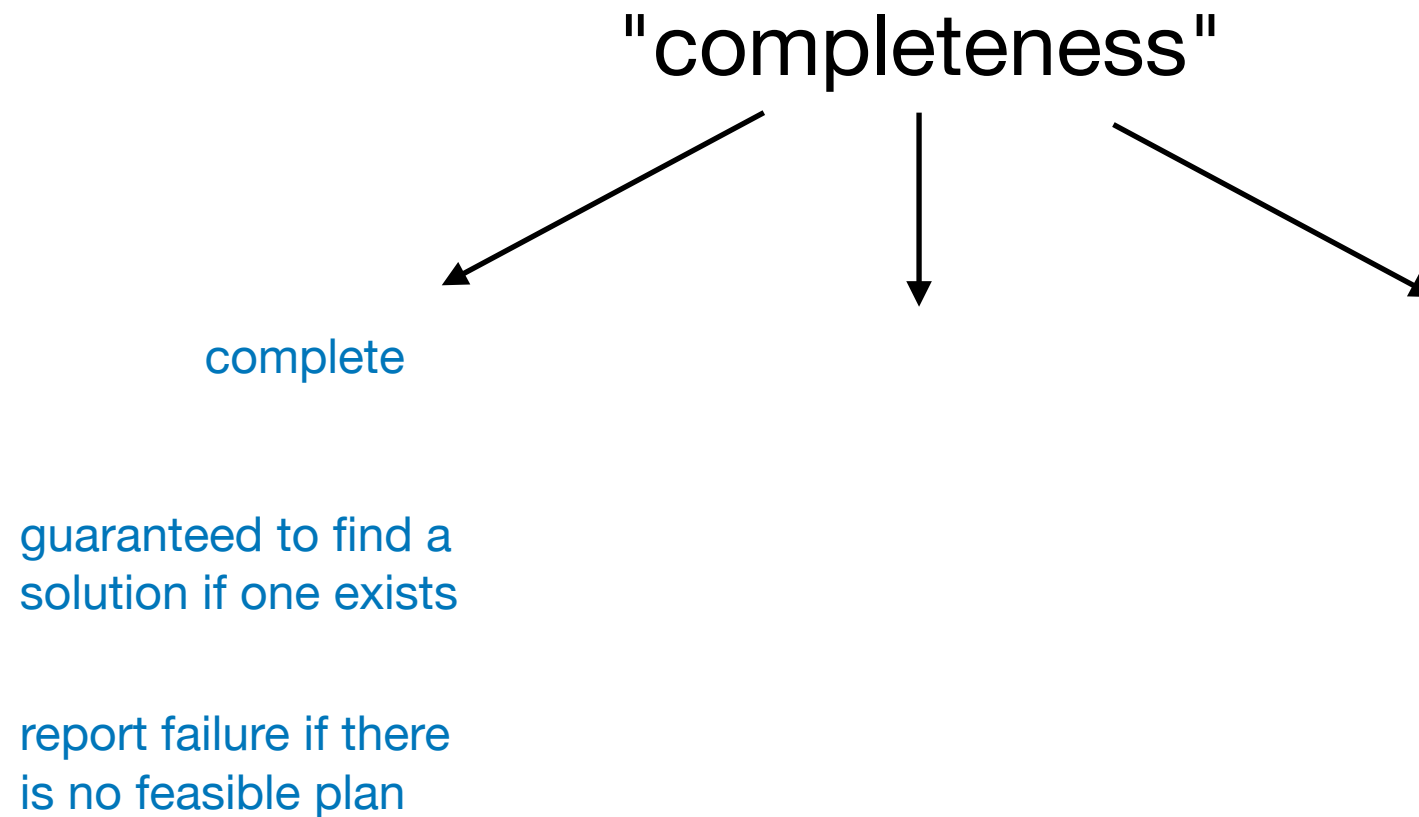


(e)



(f)

Properties of Motion Planning Problem



Properties of Motion Planning Problem

"completeness"

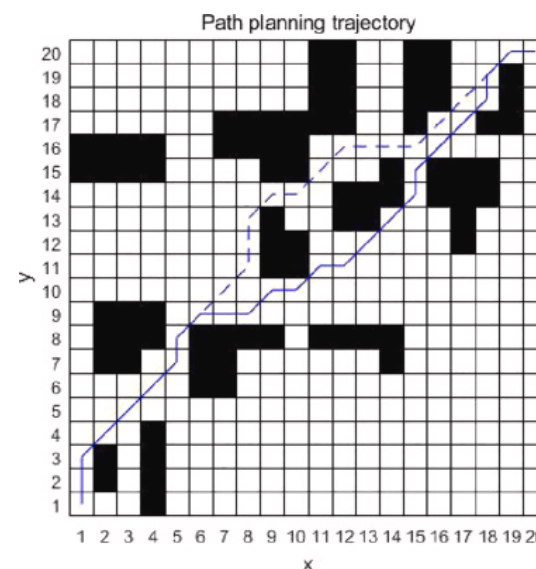
complete

guaranteed to find a solution if one exists

report failure if there is no feasible plan

resolute complete

guaranteed to find a solution if one exists at the discretised resolution



Properties of Motion Planning Problem

"completeness"

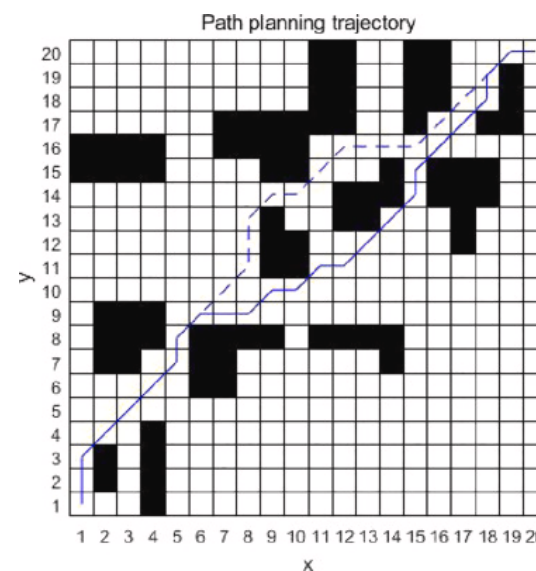
complete

guaranteed to find a solution if one exists

report failure if there is no feasible plan

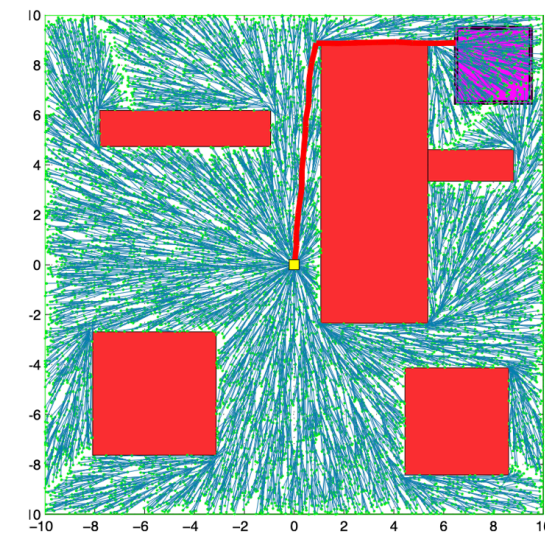
resolute complete

guaranteed to find a solution if one exists at the discretised resolution



probabilistic complete

probability of finding a solution tends to 1 as planning time goes to infinity



This Class

- What is motion planning?
- Basic motion planning problem
- Types and properties of motion planning problems
- Representation of obstacles
- Some demos

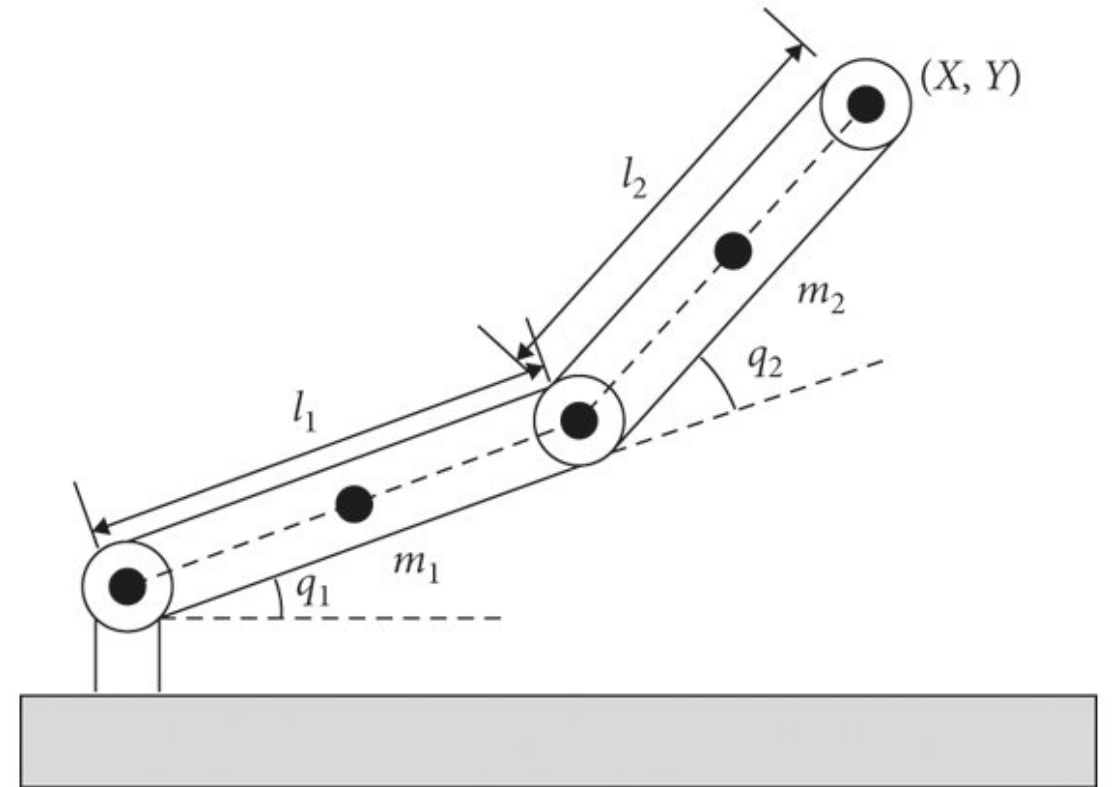
Configuration Space

- The configuration space (C-space) is the set of all possible configurations of a robot. A configuration is a complete specification of where every part of the robot is, so each configuration is a single point in C-space.
- If the robot has n degrees of freedom, C-space is usually a manifold of dimension n . A point robot in the plane has a 2D C-space. A rigid body in the plane (like the piano) has a 3D one, (x, y, θ) .

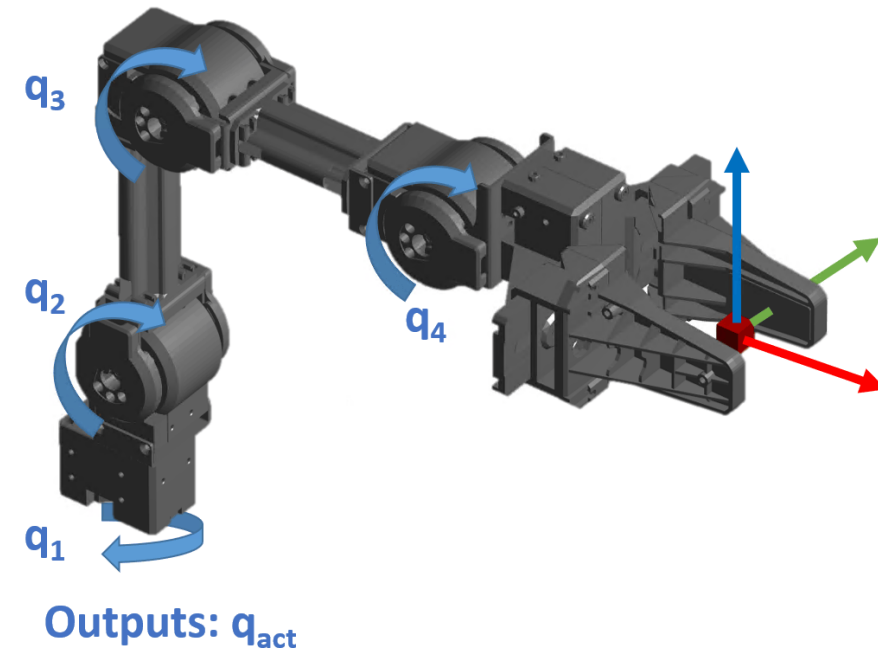
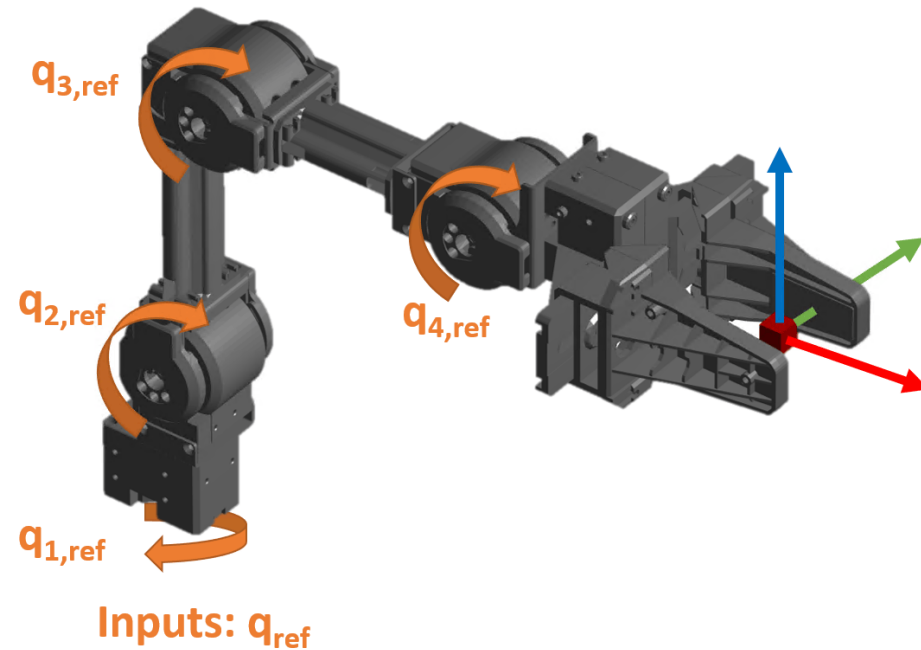
Configuration Space

$$q = [q_1, q_2]$$

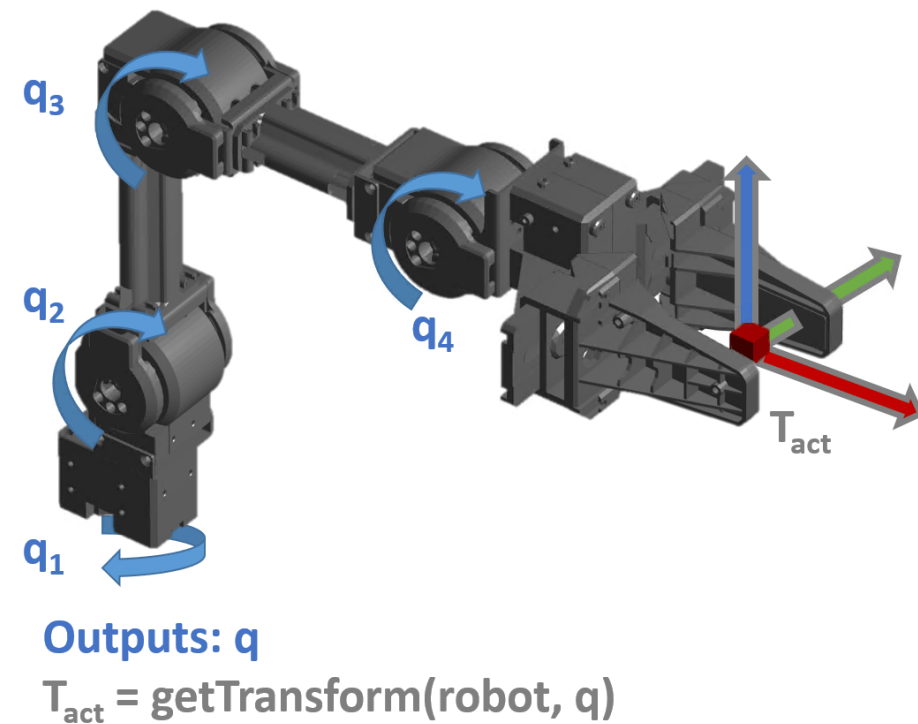
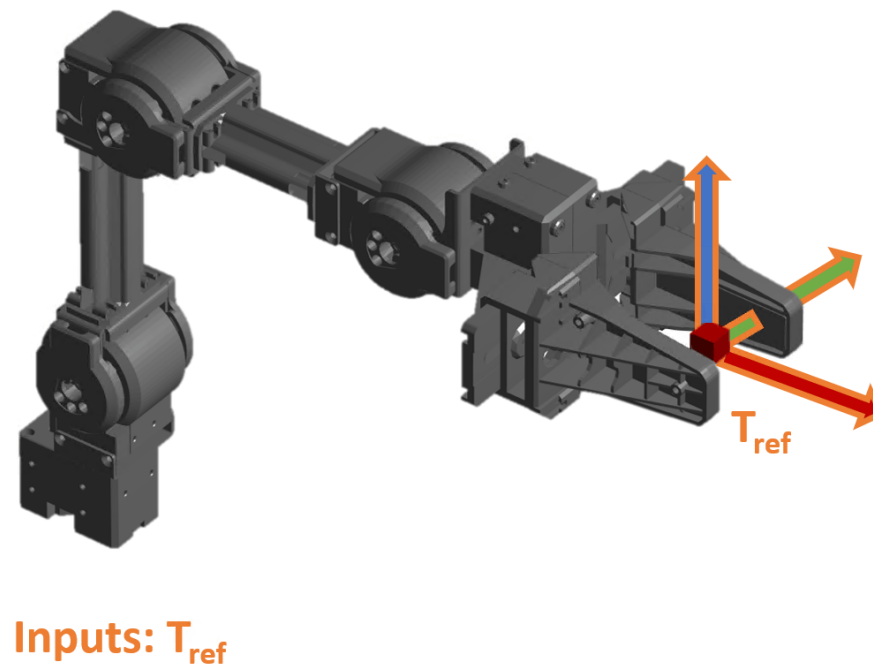
$$\mathcal{C} = S^1 \times S^1$$



C-space planning and control



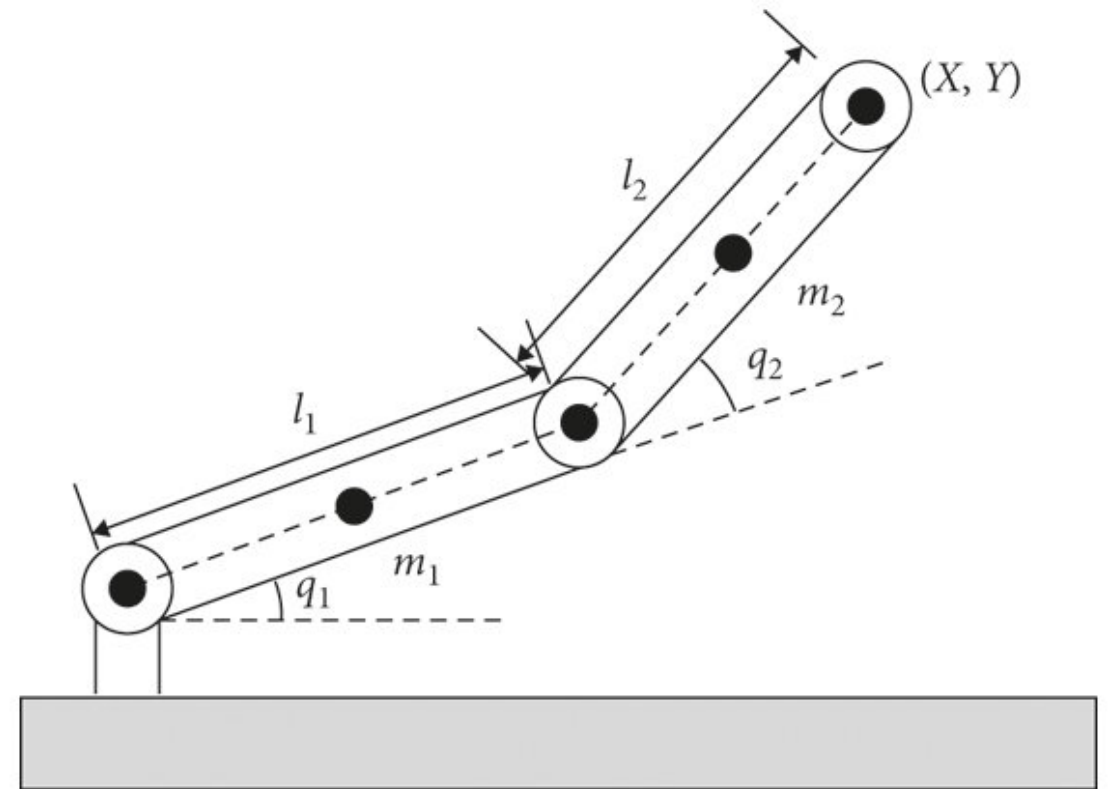
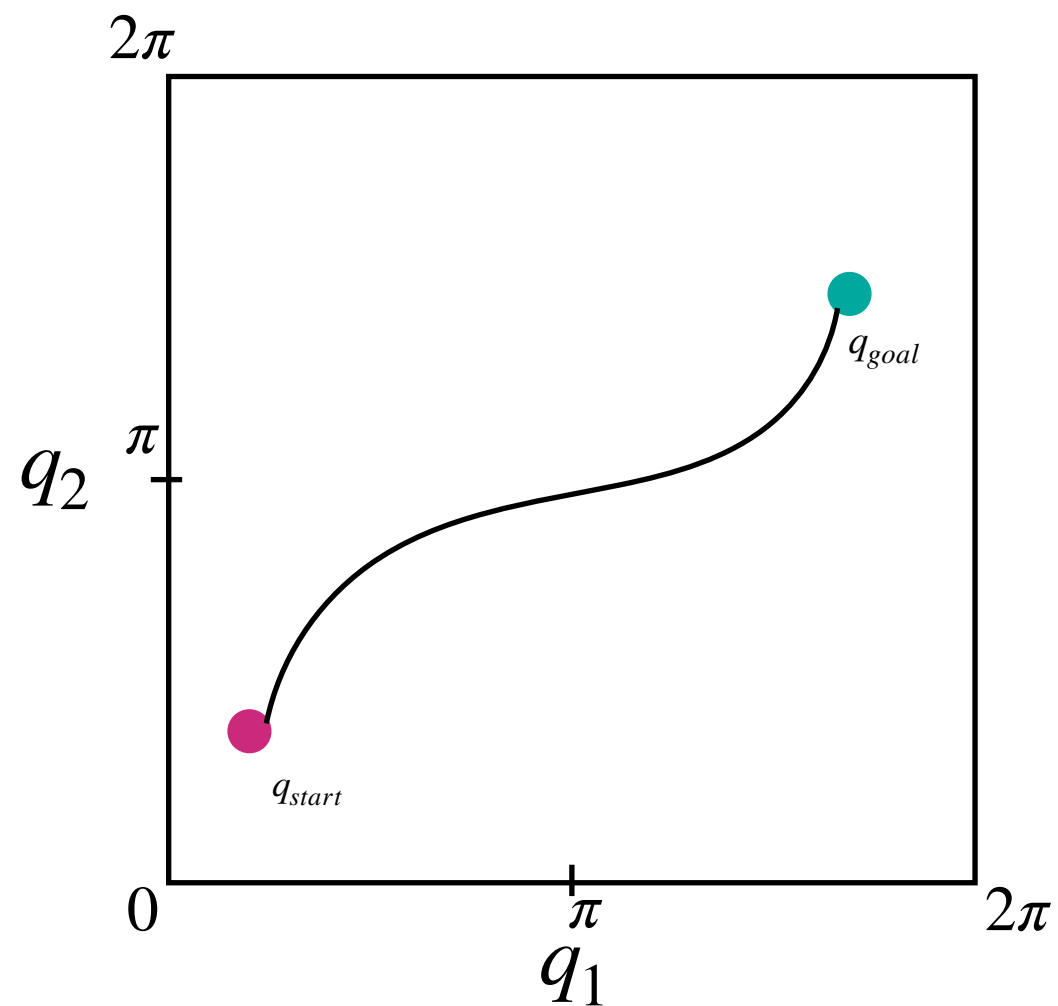
Task-space planning and control



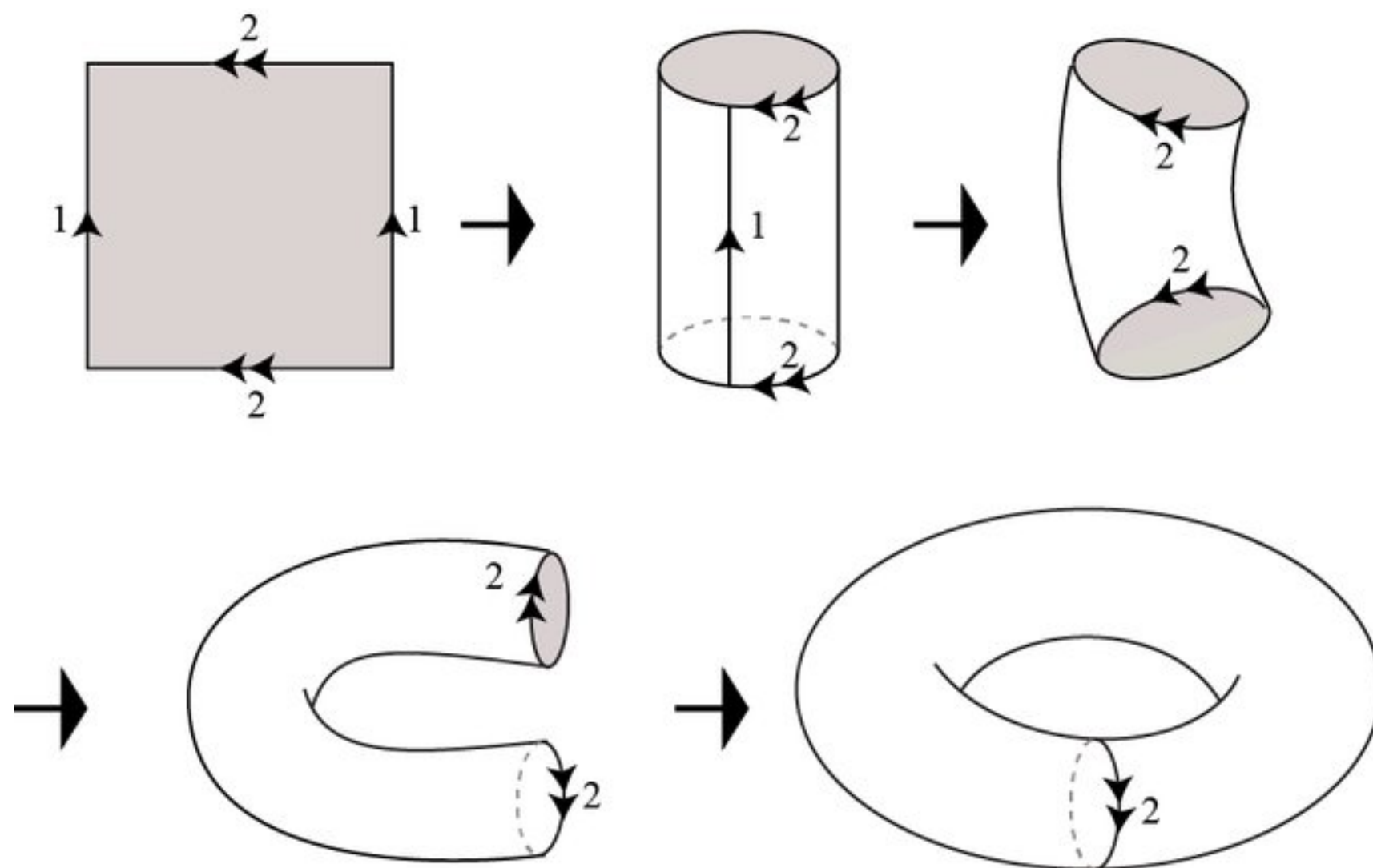
Configuration Space

$$q = [q_1, q_2]$$

$$\mathcal{C} = S^1 \times S^1$$



$$\mathcal{C} = \mathbb{S}^1 \times \mathbb{S}^1$$



Configuration Space

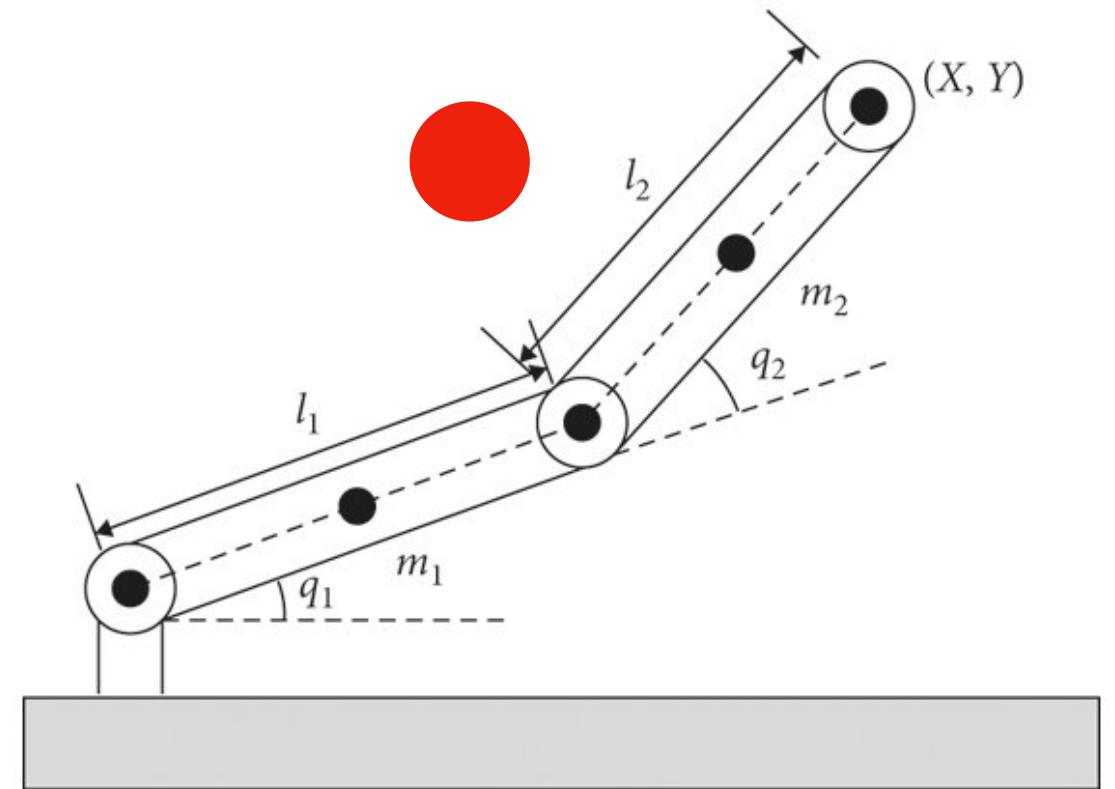
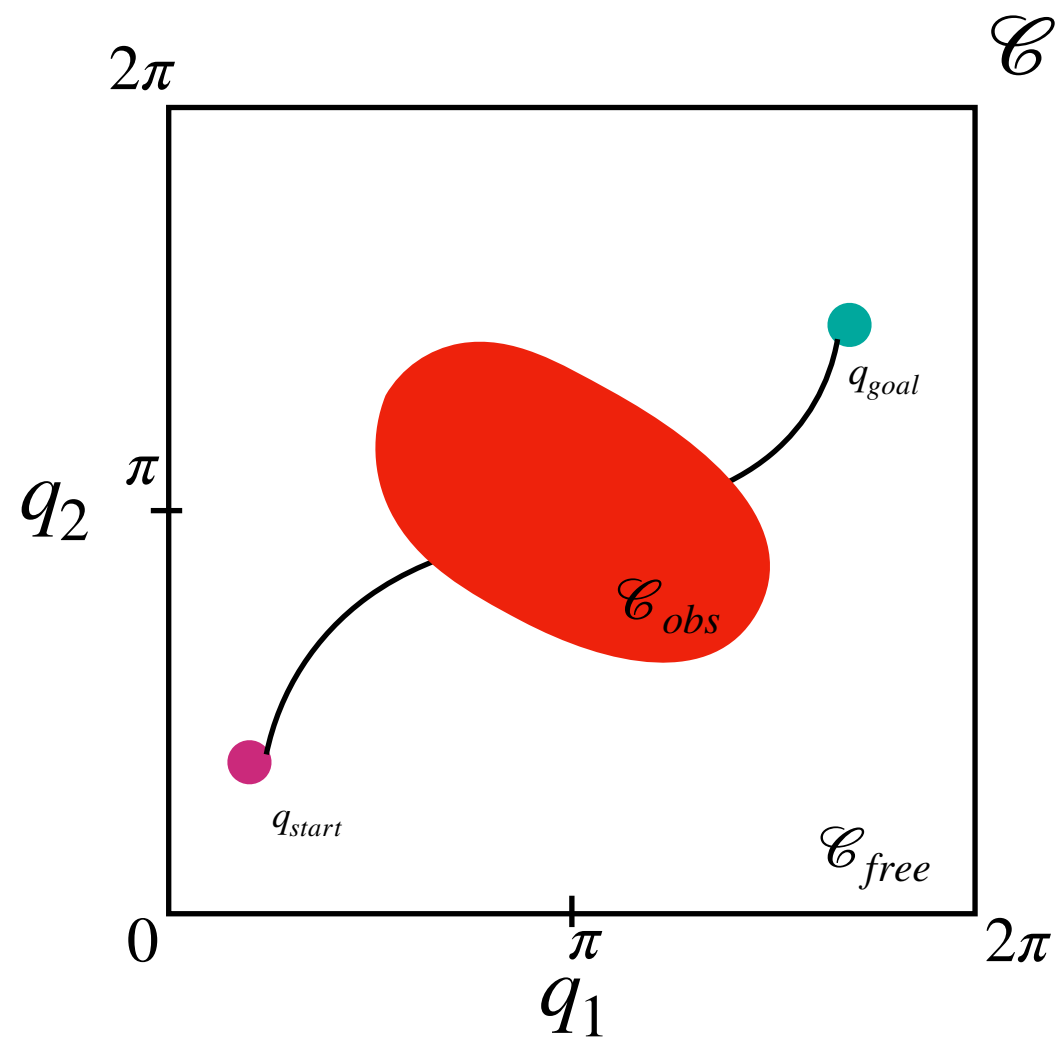
DRAKE C-space demo

DRAKE controls demo

Configuration Space

$$q = [q_1, q_2]$$

$$\mathcal{C} = \mathbb{S}^1 \times \mathbb{S}^1$$



Configuration Space

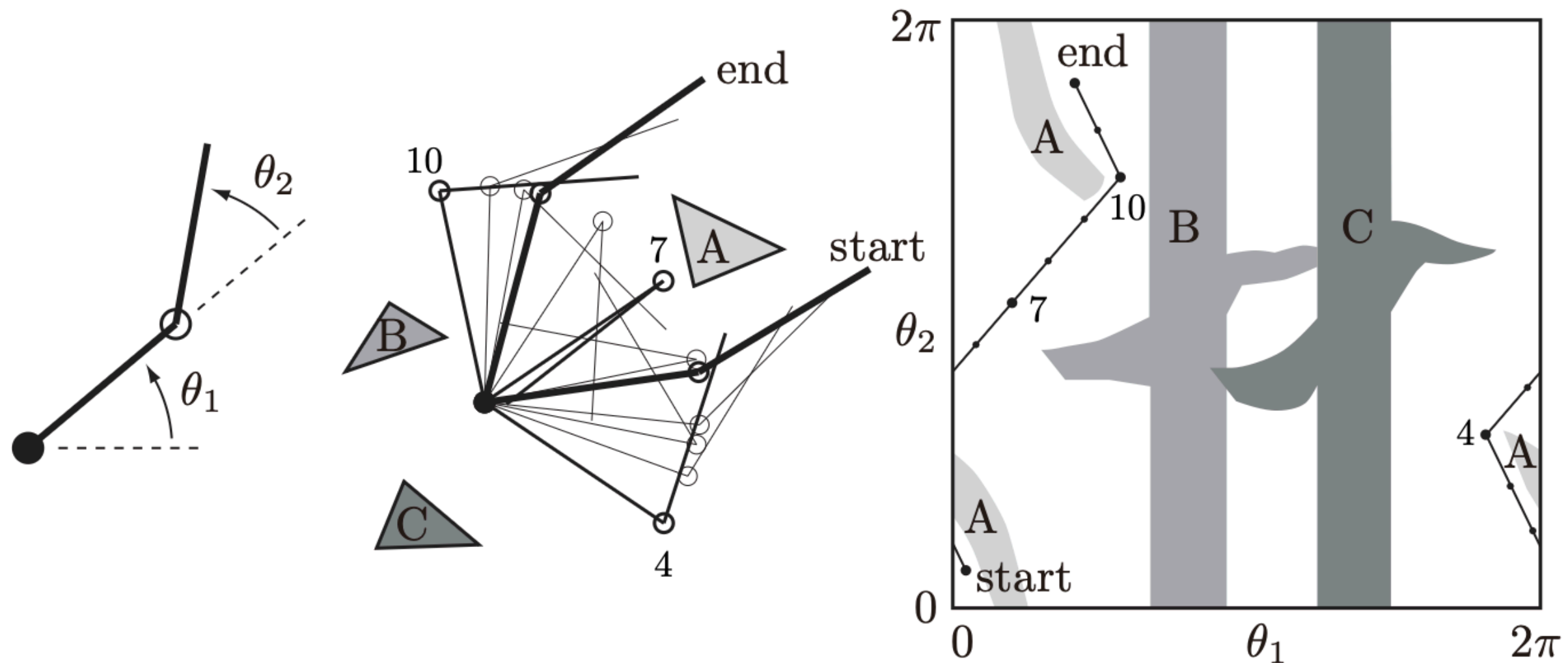


Figure 10.2: (Left) The joint angles of a 2R robot arm. (Middle) The arm navigating among obstacles A, B, and C. (Right) The same motion in C-space. Three intermediate points, 4, 7, and 10, along the path are labeled.

Configuration Space

DRAKE two link demo

Configuration Space

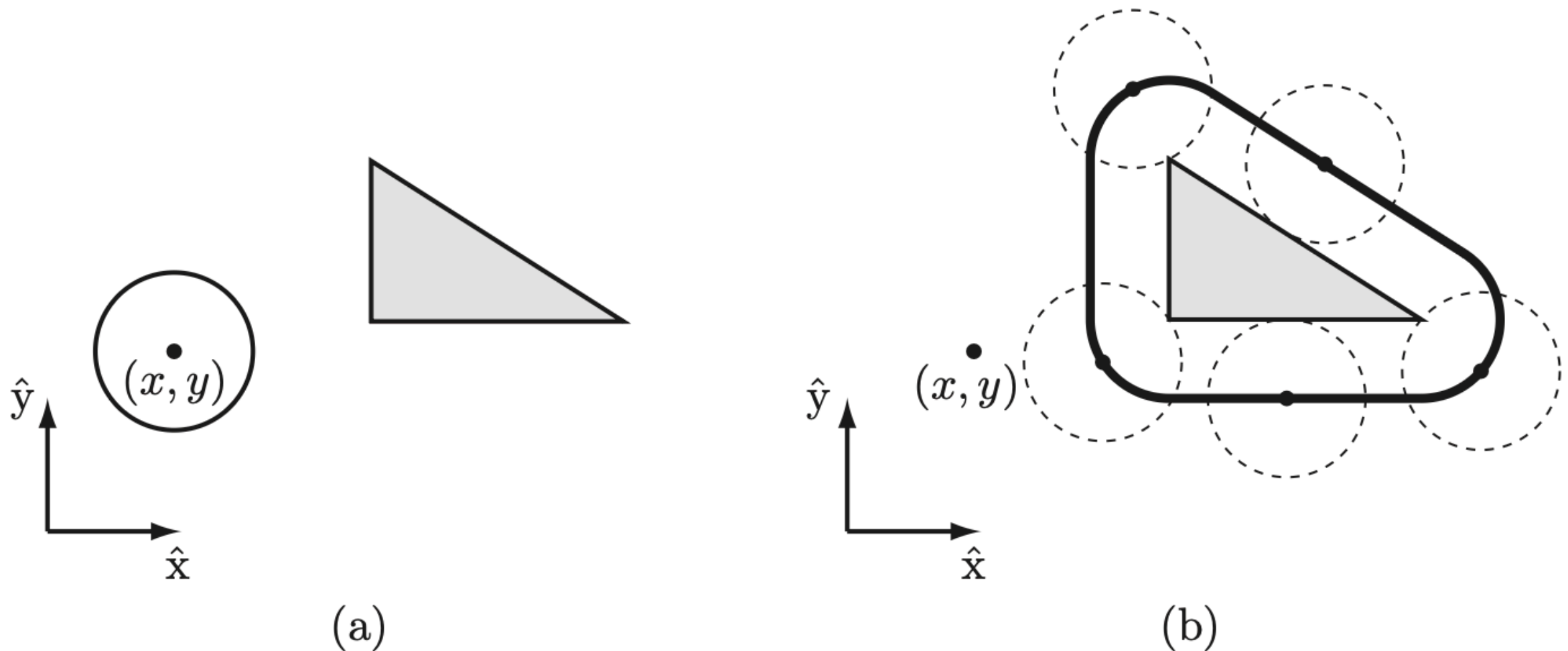


Figure 10.3: (a) A circular mobile robot (open circle) and a workspace obstacle (gray triangle). The configuration of the robot is represented by (x, y) , the center of the robot. (b) In the C-space, the obstacle is “grown” by the radius of the robot and the robot is treated as a point. Any (x, y) configuration outside the bold line is collision-free.

Configuration Space

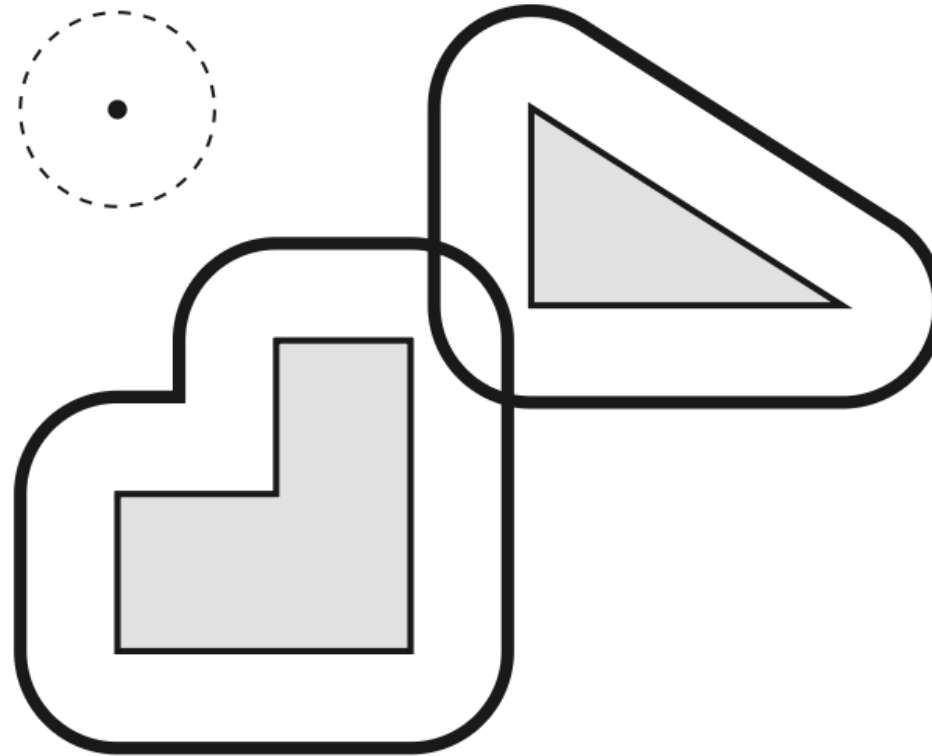


Figure 10.4: The “grown” C-space obstacles corresponding to two workspace obstacles and a circular mobile robot. The overlapping boundaries mean that the robot cannot move between the two obstacles.

Configuration Space

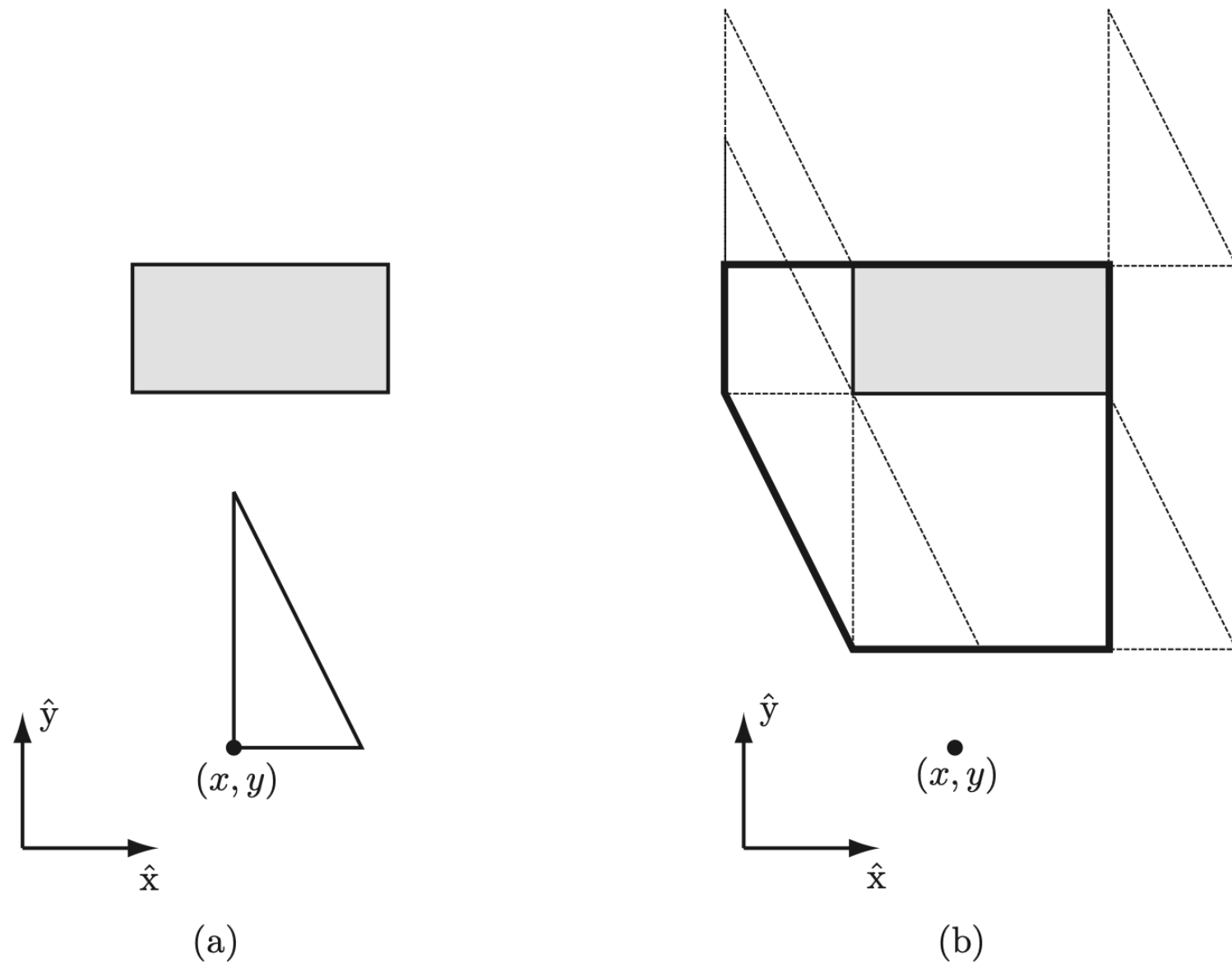


Figure 10.5: (a) The configuration of a triangular mobile robot, which can translate but not rotate, is represented by the (x, y) location of a reference point. Also shown is a workspace obstacle in gray. (b) The corresponding C-space obstacle (bold outline) is obtained by sliding the robot around the boundary of the obstacle and tracing the position of the reference point.

Configuration Space

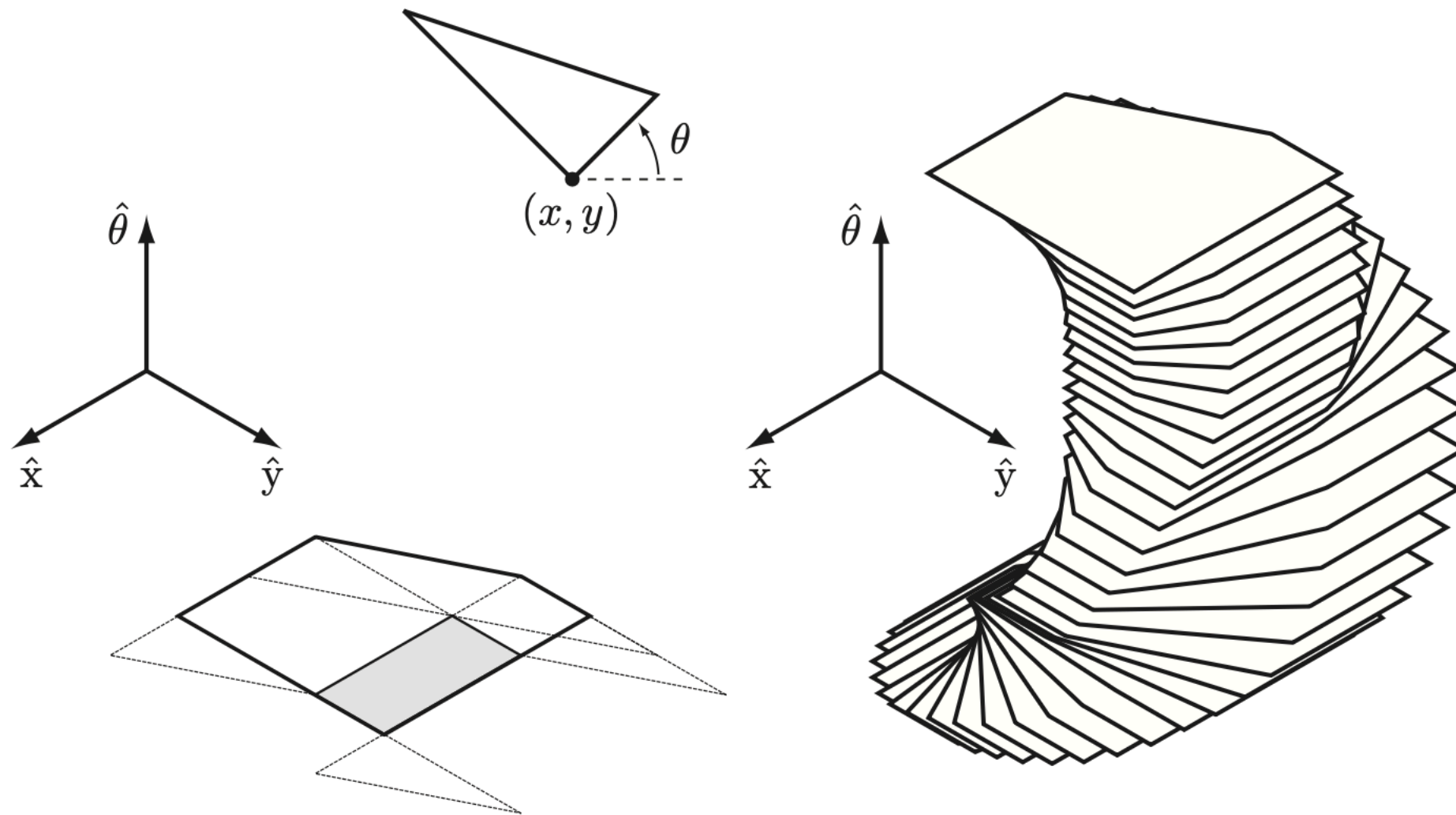
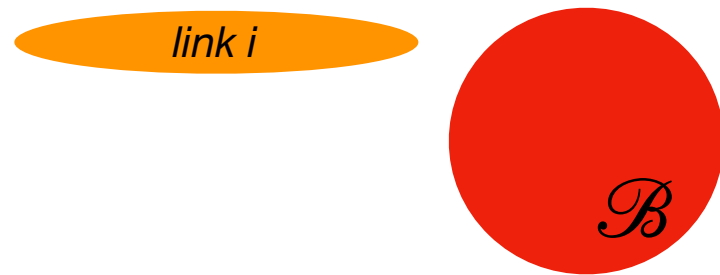


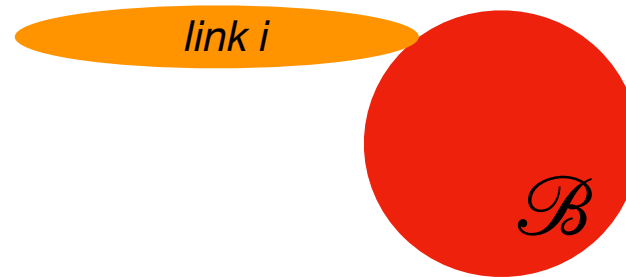
Figure 10.6: (Top) A triangular mobile robot that can both rotate and translate, represented by the configuration (x, y, θ) . (Left) The C-space obstacle from Figure 10.5(b) when the robot is restricted to $\theta = 0$. (Right) The full three-dimensional C-space obstacle shown in slices at 10° increments.

Distance to Obstacles

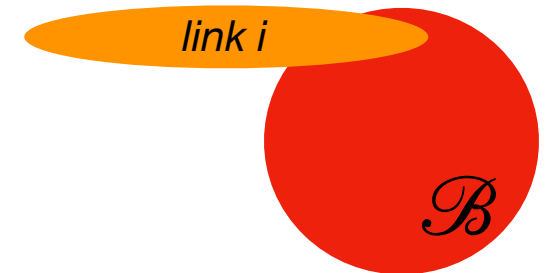
$$d(q, \mathcal{B}) > 0$$



$$d(q, \mathcal{B}) = 0$$



$$d(q, \mathcal{B}) < 0$$



- Distance between convex bodies is faster
- Gilbert-Johnson-Keerthi (GJK) algorithm efficiently computes the distance between two convex bodies
- Often enough a sphere approximation is faster

Distance to Obstacles

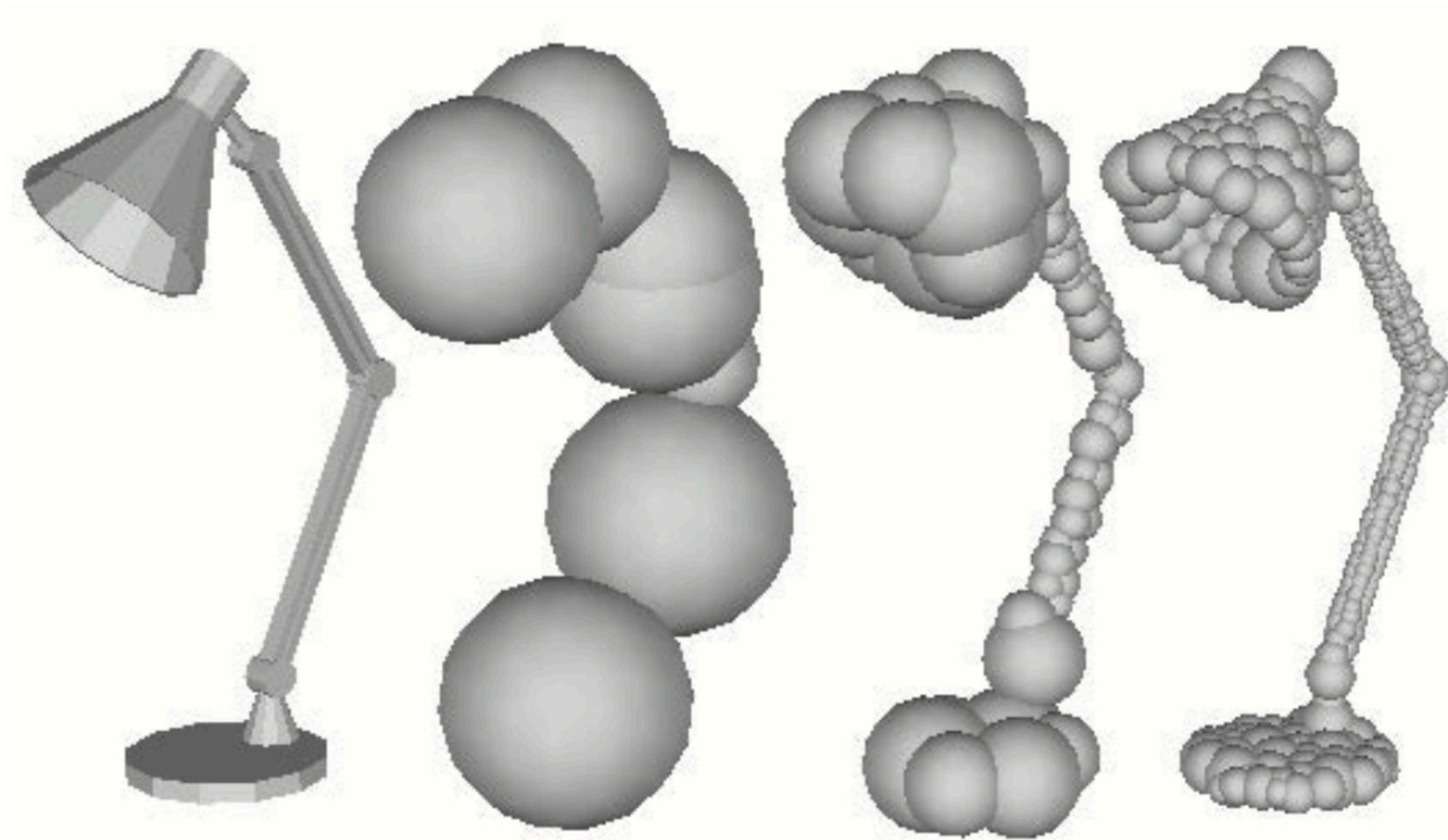
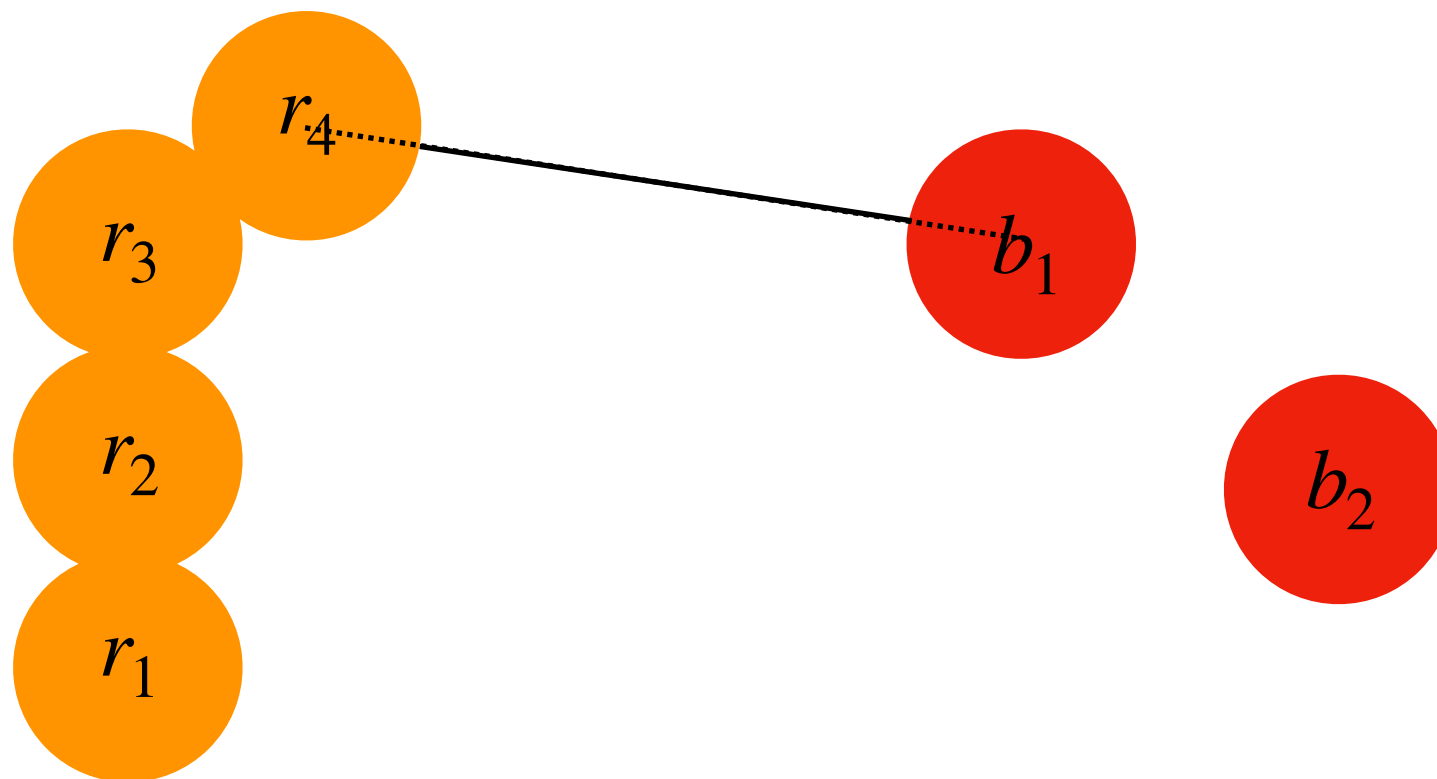


Figure 10.7: A lamp represented by spheres. The approximation improves as the number of spheres used to represent the lamp increases. Figure from [61] used with permission.

Distance to Obstacles

- Approximate the robot using k spheres with centres $r_i(q)$ and radii R_i
- Approximate the obstacles using l spheres with centres b_j and radii B_j
- Minimum separation distance

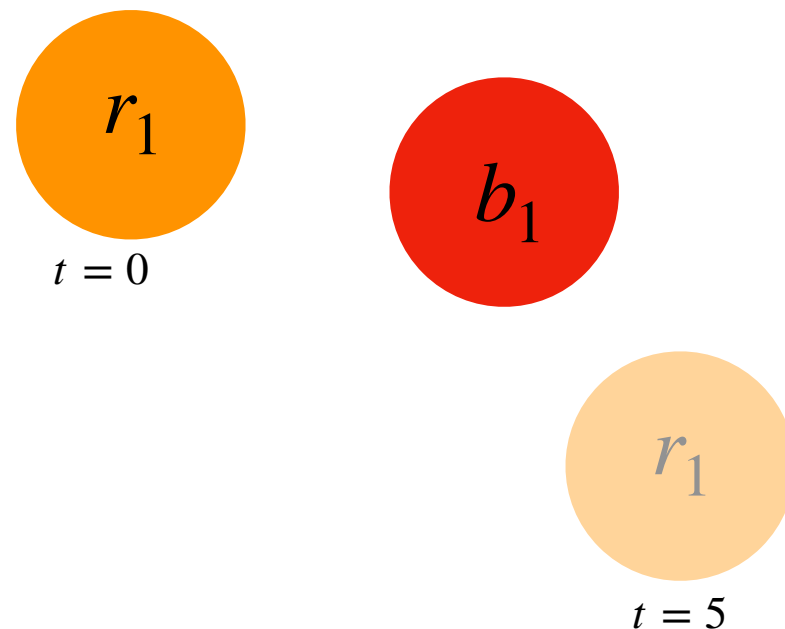
$$d(q, \mathcal{B}) = \min_{i,j} \left(\|r_i(q) - b_j\| - R_i - B_j \right)$$



Distance to Obstacles

- Approximate the robot using k spheres with centres $r_i(q)$ and radii R_i
- Approximate the obstacles using l spheres with centres b_j and radii B_j
- Minimum separation distance

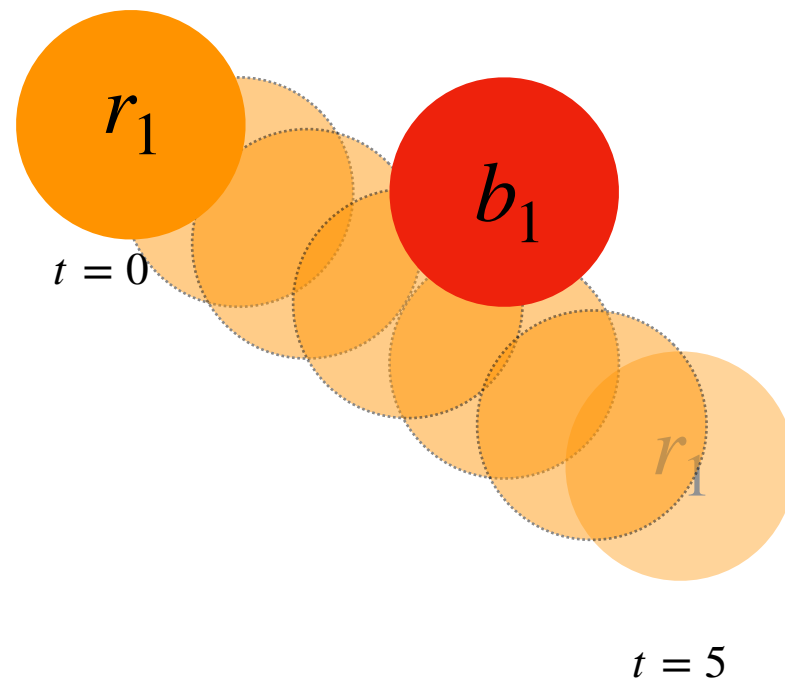
$$d(q, \mathcal{B}) = \min_{i,j} \left(\|r_i(q) - b_j\| - R_i - B_j \right)$$



Distance to Obstacles

- Approximate the robot using k spheres with centres $r_i(q)$ and radii R_i
- Approximate the obstacles using l spheres with centres b_j and radii B_j
- Minimum separation distance

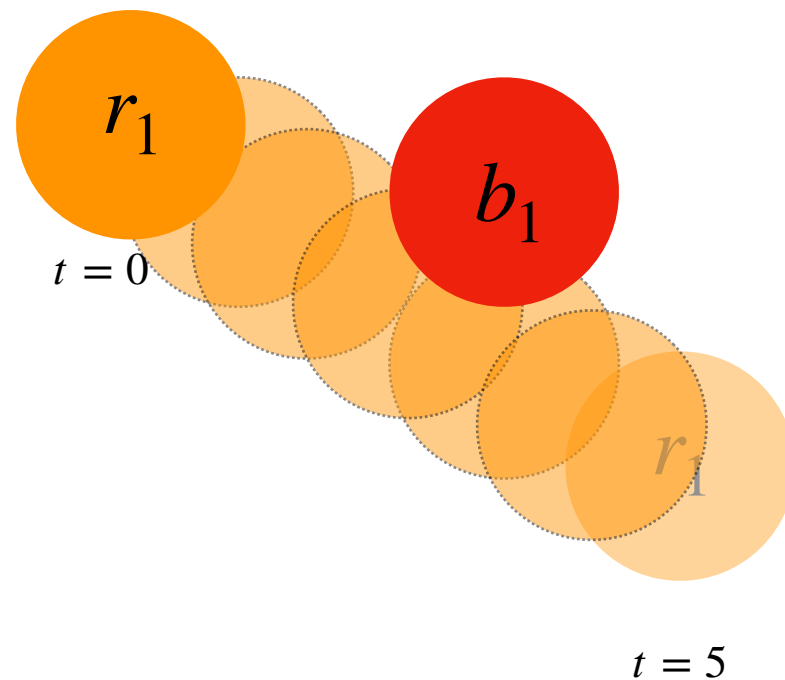
$$d(q, \mathcal{B}) = \min_{i,j} \left(\|r_i(q) - b_j\| - R_i - B_j \right)$$



Distance to Obstacles

- Approximate the robot using k spheres with centres $r_i(q)$ and radii R_i
- Approximate the obstacles using l spheres with centres b_j and radii B_j
- Minimum separation distance

$$d(q, \mathcal{B}) = \min_{i,j} \left(\|r_i(q) - b_j\| - R_i - B_j \right)$$



Distance to Obstacles

DRAKE collision demo

Overview of Motion Planning Algorithms

Approach	Key idea	Examples
Geometric / Combinatorial	Construct an explicit representation of free space and search for a path.	Cell decomposition, Visibility graphs
Graph-based Search	Discretise C-space into a graph and search for a feasible or shortest path.	Dijkstra, A*
Sampling-based	Explore C-space by sampling configurations and connecting collision-free states.	PRM, RRT, RRT*
Optimisation-based	Iteratively improve an initial trajectory to minimise cost while satisfying constraints.	CHOMP, TrajOpt
Optimal Control	Plan trajectories using system dynamics, control inputs and an objective function.	MPC, MPPI, Direct collocation
Reactive / Potential Fields	Generate local motion commands using attractive and repulsive objectives.	Artificial Potential Fields, DWA
Learning-based / Generative	Learn motion patterns or generate trajectories from data.	Diffusion planning, Neural motion planners