ECE 470 Recap and Review
Trajectory Generation and Motion Planning

Prof. Katie Driggs-Campbell

Contact: krdc@Illinois.edu

Notes from Modern Robotics Chapter 9 and 10

Administrivia

- Extra credit (up to 1% of grade) will be given for participating in the zoom quizzes
- Exercises associated with lectures on PrairieLearn
- Online Office Hours on **Wednesday 10:30am China Time** (Tuesday 9:30pm Illinois Time)

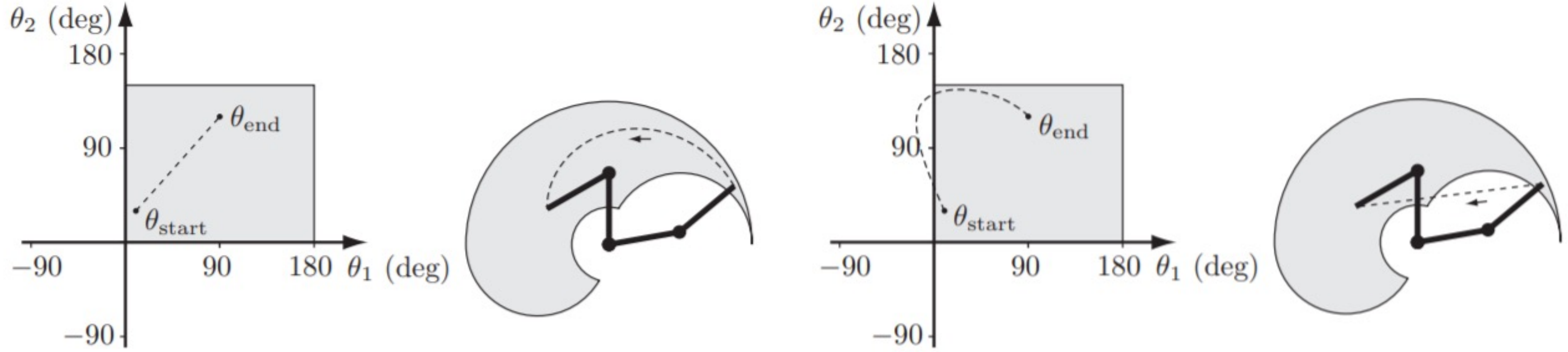
Lecture Goals

- Learned the basics of **trajectory generation and motion planning**
- For **trajectory generation**, supposing we are given a geometric path, we learned about how time scaling can help us meet requirements
- **Motion planning** provides us with tools to find a trajectory that can reach a goal without collisions

Trajectory Generation

Summary of Trajectory Generation Lecture

- A trajectory is a geometric path (set of points) and a specification of timing (time scaling)
  - Discussed the notion of a "straight line" path in different spaces
- We introduce the idea of normalized paths ($\theta\colon [0,1] \rightarrow \Theta$) and use a time-scaling function to define the timing of the trajectory ($s\colon [0,T] \rightarrow [0,1]$)



Steps for Trajectory Generation

- Step 1: Identify requirements and limits of your system
- Step 2: Determine the geometric path
- Step 3: Pick parameterization for time-scaling
- Step 4: Solve for $s(t)$ to meet requirements

Time-Scaling Parameterization

Time-Scaling Parameterization

1. $\theta(t) = \theta_0 + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\theta_1 - \theta_0)$
2. $\dot{\theta}(t) = \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_1 - \theta_0)$
3. $\ddot{\theta}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\theta_1 - \theta_0)$

Time-Scaling Parameterization

1. $\theta(t) = \theta_0 + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\theta_1 - \theta_0)$
2. $\dot{\theta}(t) = \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_1 - \theta_0)$
3. $\ddot{\theta}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\theta_1 - \theta_0)$

- Find max velocity and tune to meet requirements

$\rightarrow \dot{\theta}_{max} = \frac{3}{2T}(\theta_1 - \theta_0)$

- Find max acceleration and tune to meet requirements

$\rightarrow \ddot{\theta}_{max} = \pm\frac{6}{T^2}(\theta_1 - \theta_0)$

Double check your answers!

Trajectory generation

We want a robot to move from one location to another location in a straight line with time $T = 2$. Let's parameterize the path as follows:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

where time $t \in [0, T]$. We need the robot to start with zero velocity, and reach the destination with zero velocity. Find a_0, a_1, a_2, a_3.

$a_0 =$	number (rtol=0.01, atol=1e-08)	?
$a_1 =$	number (rtol=0.01, atol=1e-08)	?
$a_2 =$	number (rtol=0.01, atol=1e-08)	?
$a_3 =$	number (rtol=0.01, atol=1e-08)	?

Quick Recap

- We choose a **parametrization** $\boldsymbol{s(t)}$, and computed the resulting velocity and acceleration profiles of the trajectory
  - Using a third-order polynomial, we tuned their maximal values to meet requirements with one parameter T

Quick Recap

- We choose a **parametrization** $\boldsymbol{s(t)}$, and computed the resulting velocity and acceleration profiles of the trajectory
  - Using a third-order polynomial, we tuned their maximal values to meet requirements with one parameter T
- We can follow the same procedure with different parametrizations for $s(t)$ (e.g., polynomials of order 5, trapezoidal functions, splines, etc.)
  - Having more parameters allows us to meet more constraints. For example, using a fifth order polynomial, we can ensure that $\ddot{\theta}(0) = \ddot{\theta}(T) = 0$, meaning no jerk at beginning and end of the motion

Motion Planning

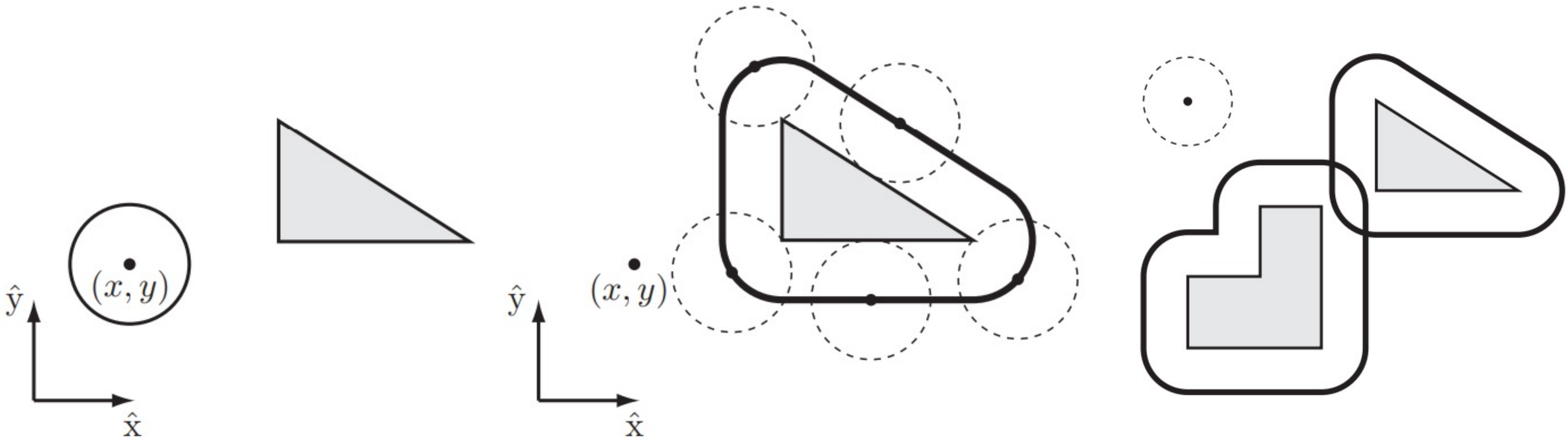
Summary of Motion Planning

Given an initial state $x(0) = x_{start}$ and a desired final state x_{goal}, find a time T and a set of controls u: $[0, T] \to \mathcal{U}$ such that the motion satisfies $x(T) = x_{goal}$ and $q\big(x(t)\big) \in \mathcal{C}_{\mathrm{free}}$ for all $t \in [0, T]$

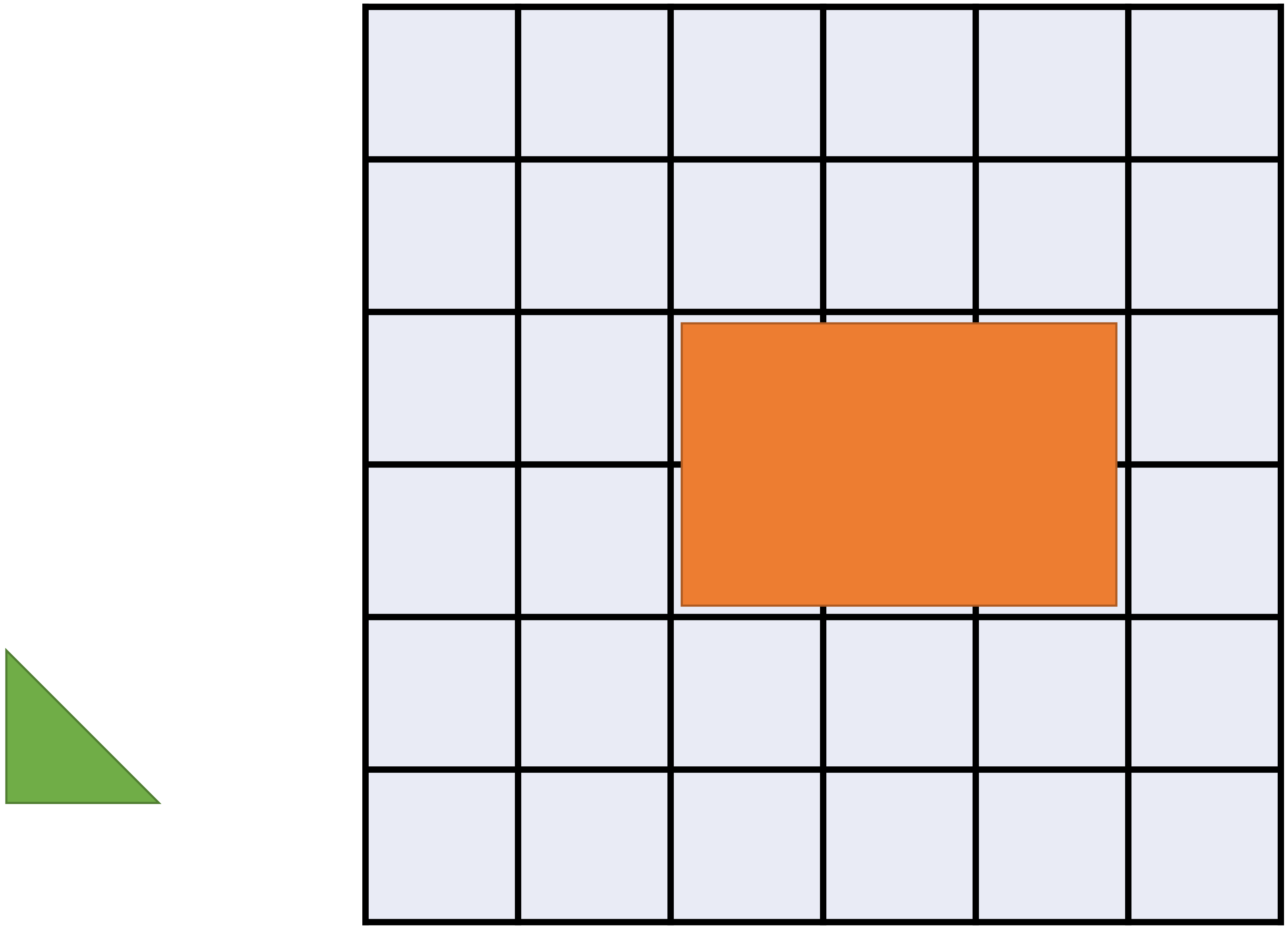
Assumptions:

1. A feedback controller can ensure that the planned motion is followed closely
2. An accurate model of the robot and environment will evaluate $\mathcal{C}_{\mathrm{free}}$ during motion planning

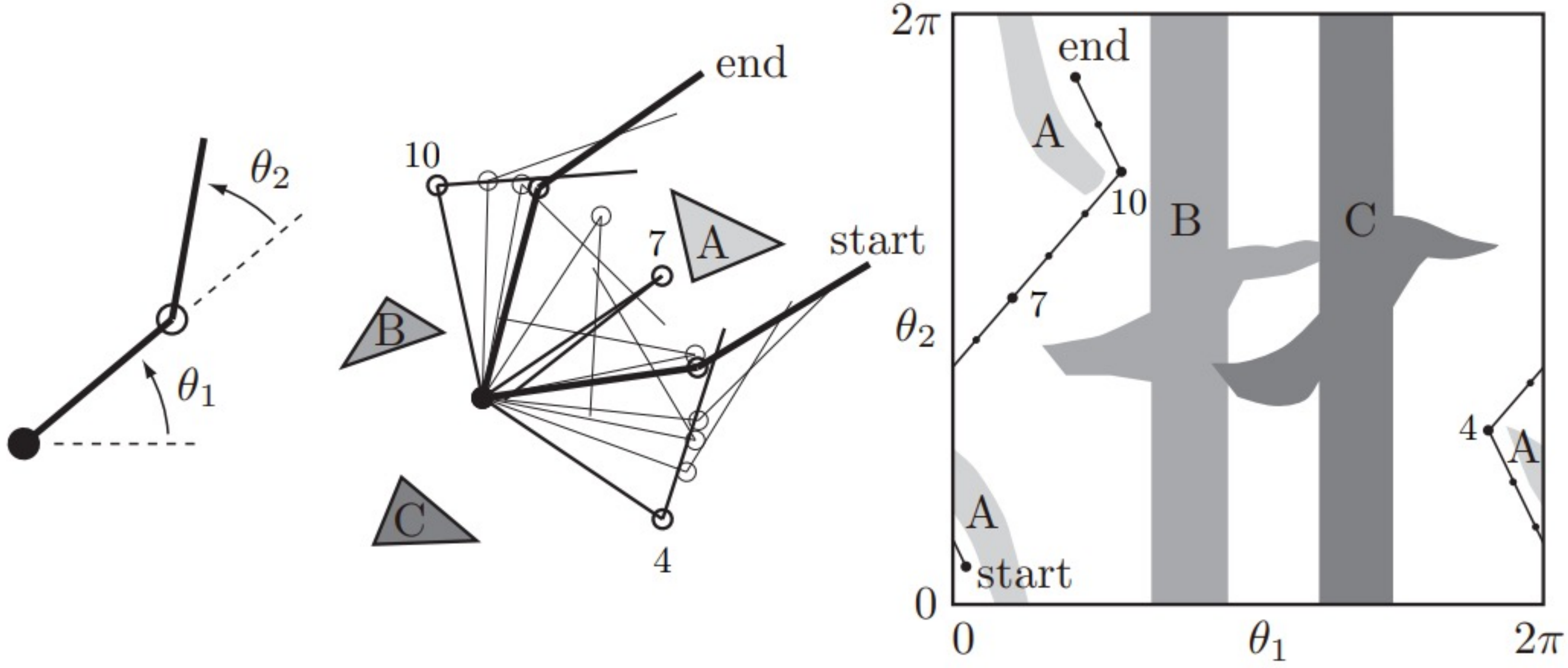
Finding Free Configuration Space



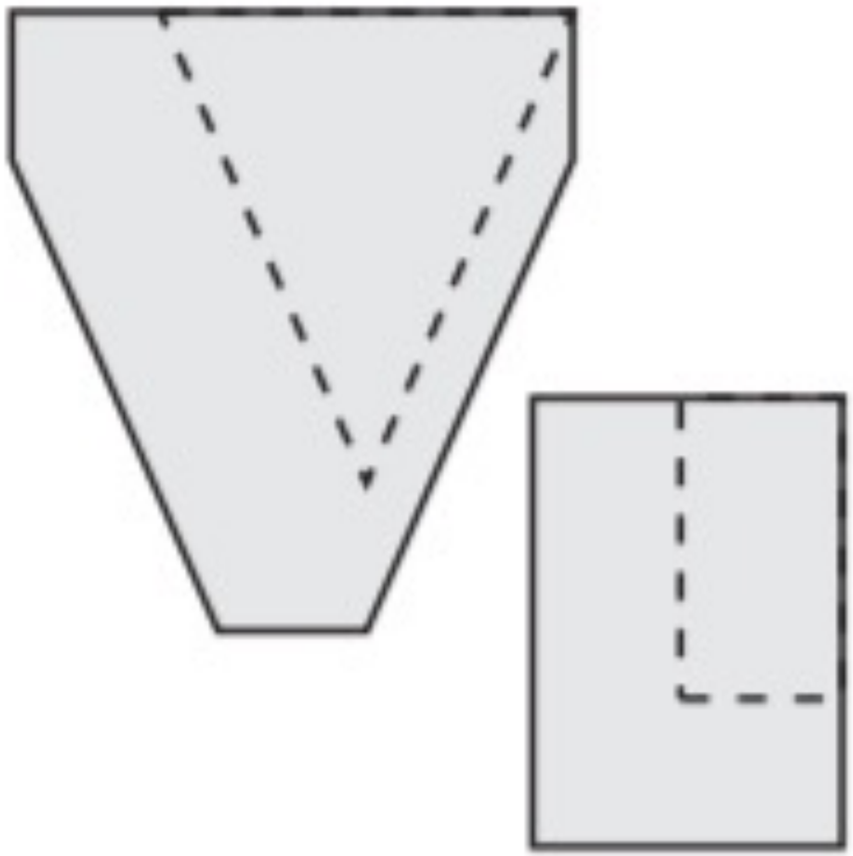
Finding Free Configuration Space



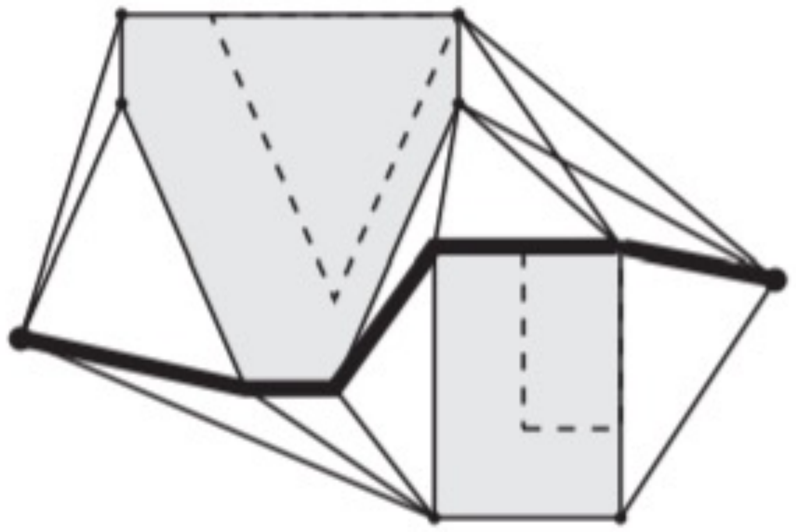
Configuration Space: 2R Planar Arm

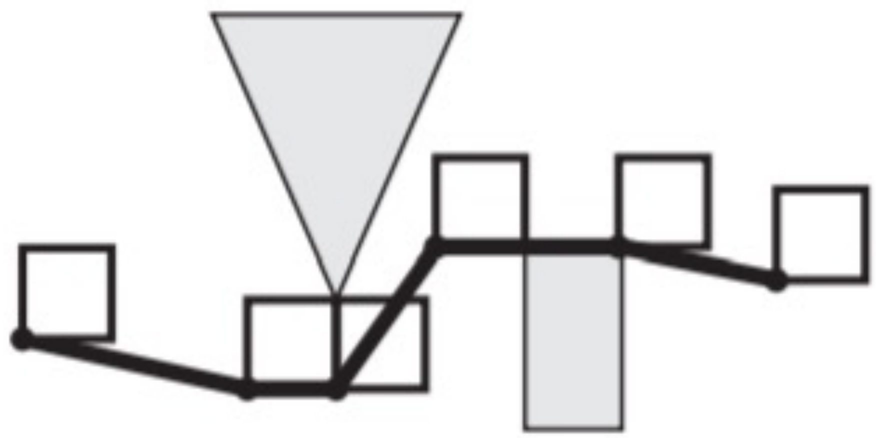


A simple roadmap: visibility graph



A simple roadmap: visibility graph





Probabilistic Roadmaps (PRMs)

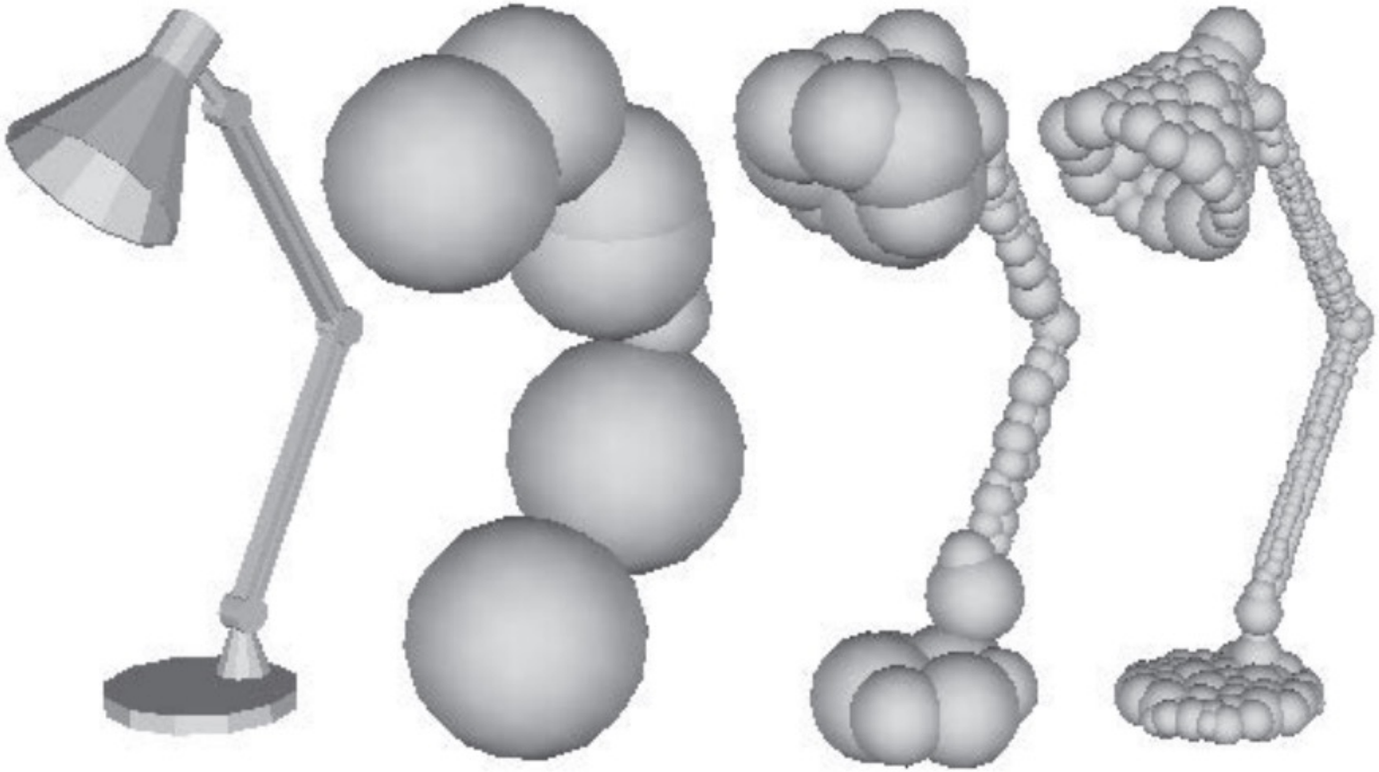
Validity Check: Collision Detection

Given a $\mathcal{C}_{obs}$ (denoted $\mathcal{B}$) and configuration q, let $d(q, \mathcal{B})$ be a distance function between the robot configuration and obstacle

- $d(q, \mathcal{B}) > 0$ means no contact
- $d(q, \mathcal{B}) = 0$ means contact
- $d(q, \mathcal{B}) < 0$ means penetration

Spherical Approximation

- One simple method is to approximate the robot and obstacles as unions of overlapping spheres
- Approximations must be **conservative**



Collision Detection for Spherical Approximation

Given a robot at q represented by k spheres of radius R_i centered at $r_i(q)$, and an obstacle $\mathcal{B}$ represented by l spheres of radius B_j centered at b_j, the distance can be calculated as:

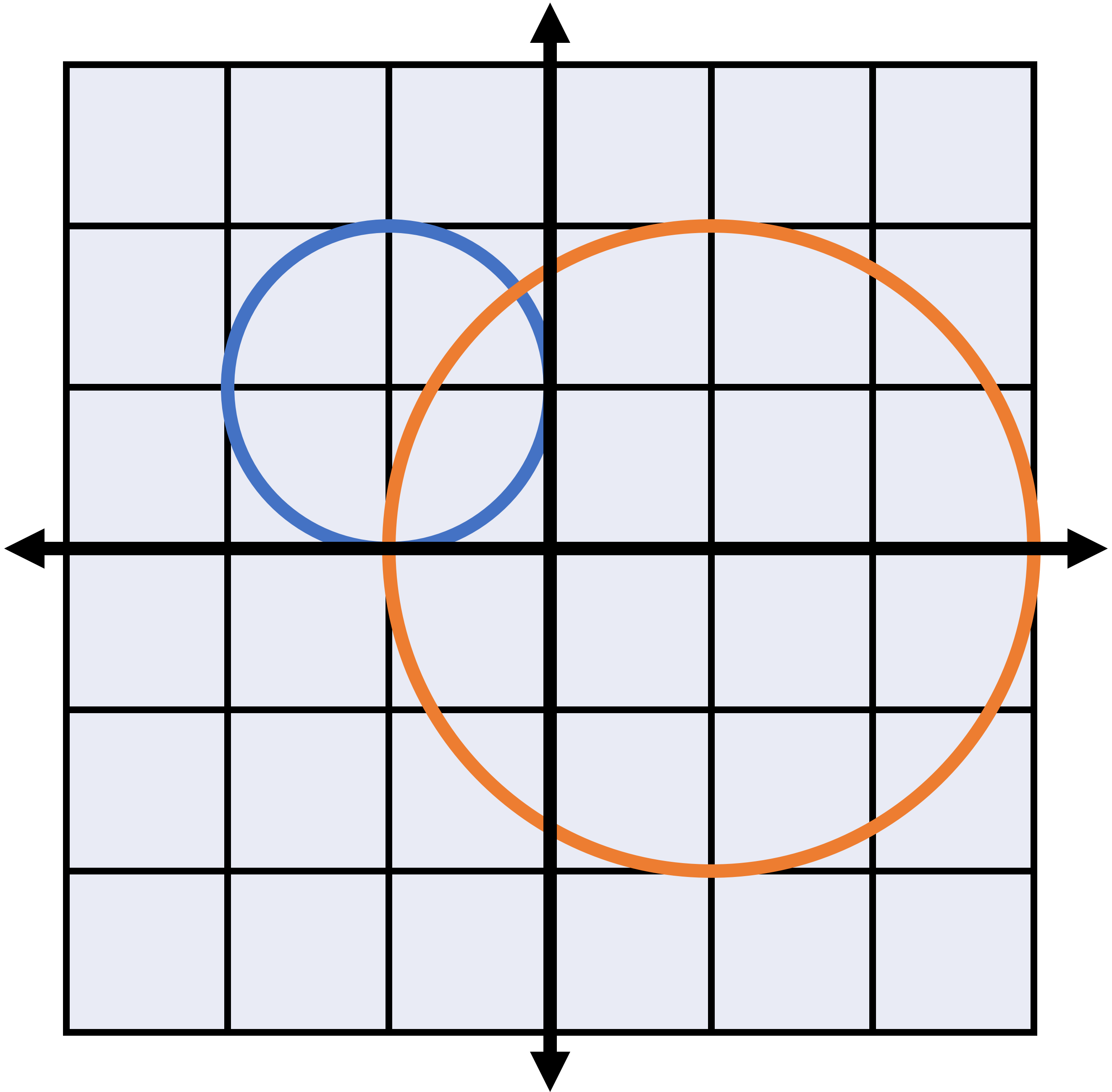
$$d(q, \mathcal{B}) = \min_{i,j} \left\| r_i(q) - b_j \right\| - R_i - \mathcal{B}_j$$

Example of collision detection (1)

Two spheres of radius r_1 and r_2 have centers at p_1 and p_2 (both in the same frame), respectively. Are these spheres in collision?

$$p_1 = (-1,1,0), r_1 = 1 \qquad\qquad p_2 = (1,0,0), r_2 = 2$$

Example of collision detection (2)



Motion Planning Summary

- Given an initial state and a desired final state, **motion planning** provides us with tools to find a time horizon and a sequence of actions to find a trajectory that reaches the goal without collisions
  - Need collision detection
- A **roadmap** path planner uses a graph representation of free space, which can then provide a trajectory using search algorithms
  - Example planners include **Visibility** and **Probabilistic Roadmaps**
  - Use your favorite graph search algorithm to determine the trajectory

Lecture Recap

- **Trajectory Generation** is often used for automation, when the path is easy to define
- **Motion Planning** allows us to *find* collision-free paths (trajectories) in high-dimensional spaces
- PrairieLearn assignment due next week!
- Office hours on Wednesday