Lecture 16
Motion Planning I

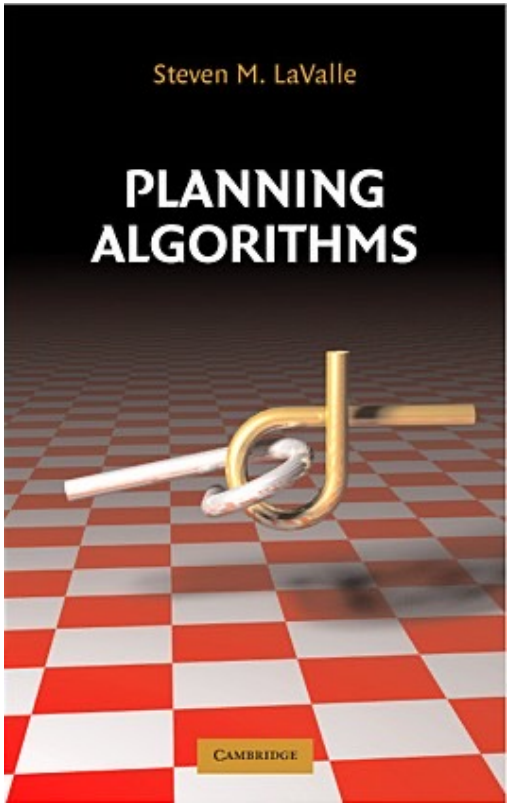
Prof. Katie DC

April 9, 2026

Modern Robotics 10.1-10.2

Administrivia

- No HW this week
- Exam 2 Velocity Kinematics Question will be extra credit



CSL Prof. LaValle central to Oculus' $2 billion success

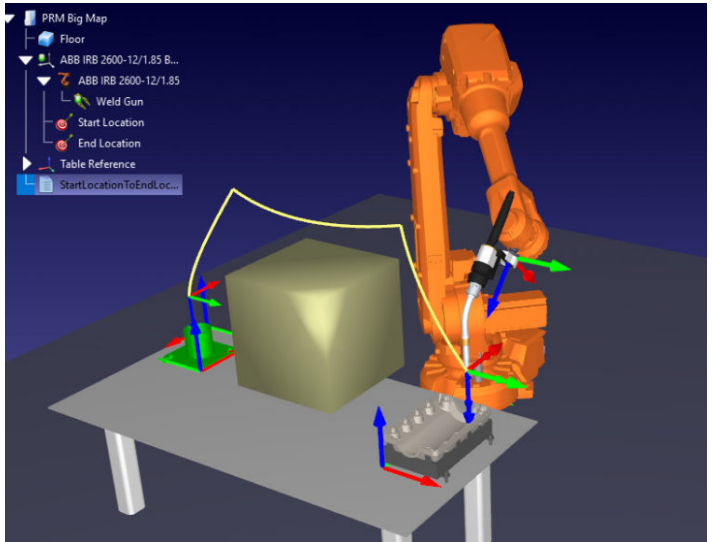
Apr 17, 2014

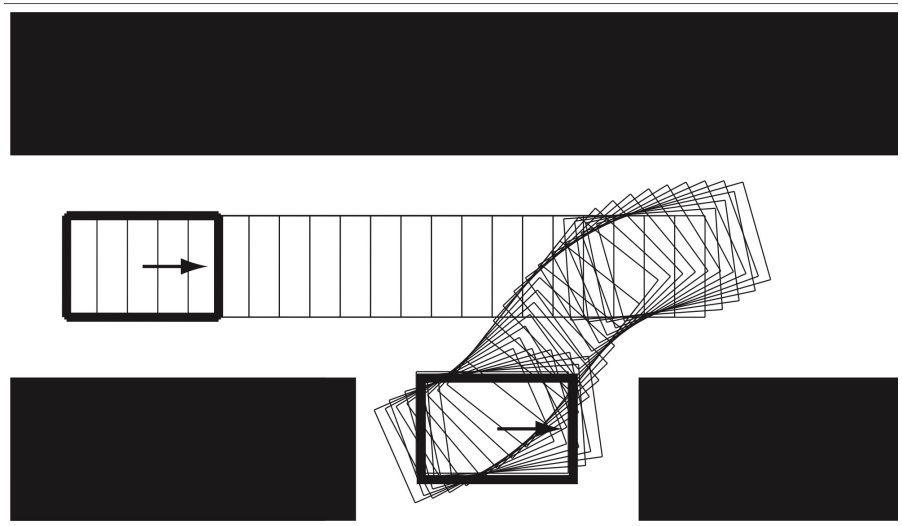
Rick Kubetz, Engineering Communications Office

On March 25, both the business and technology news pages excitedly announced Facebook's $2 billion acquisition of Oculus VR, the maker of a virtual reality gaming headset called Oculus Rift.

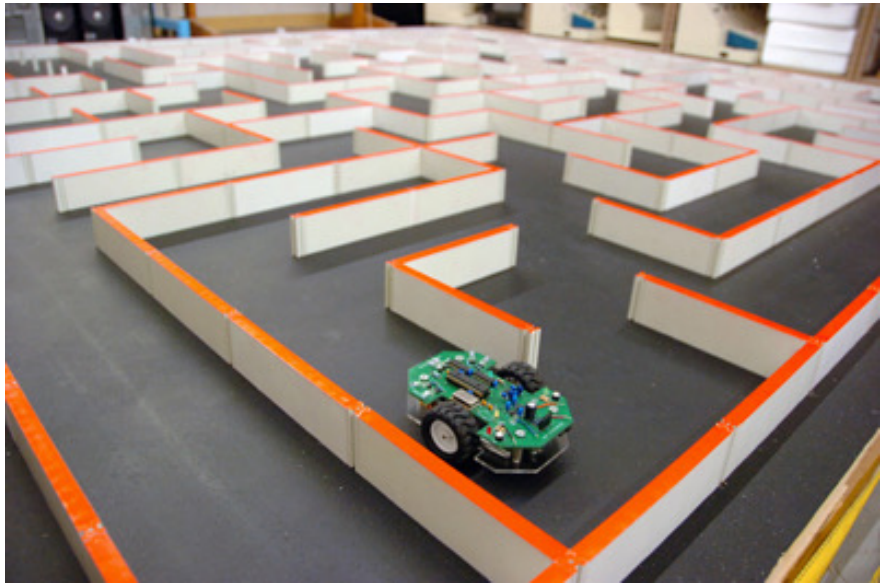
Overview of Motion Planning

- **Motion planning** is the problem of finding a robot motion from start state to a goal state that avoids obstacles in the environment
- Recall the **configuration space or C-space:** every point in the C-space $\mathcal{C} \subset \mathbb{R}^n$ corresponds to a unique configuration q of the robot
  - E.g., configuration of a robot arm is $q = (\theta_1, \theta_2, \ldots, \theta_n)$
- The **free C-space** $\mathcal{C}_{\text{free}}$ consists of the configurations where the robot neither collides with obstacles nor violates constraints





“Space” and “obstacles”



“Obstacles” -- objects robots cannot collide with; joint limits of the robots

C_{obs}

Equations of Motion for Motion Planning

control inputs: $u \in \mathcal{U} \subset \mathbb{R}^m$

state: $x = (q, v = \dot{q})$

$\mathcal{X}_{free}$: $\{x \mid q(x) \in \mathcal{C}_{free}\}$

---

equation of motion:

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

$$x(T) = x(0) + \int_0^T f(x(t), u(t))\,dt$$

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 $u: [0, T] \rightarrow \mathcal{U}$ such that the motion satisfies $x(T) = x_{goal}$ and $q\big(x(t)\big) \in \mathcal{C}_{\text{free}}$ 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 $\mathcal{C}_{\text{free}}$ during motion planning

Types of Motion Planning Problems

- **Path planning versus motion planning**
  - Trajectory generation versus these lectures
- **Control inputs: $m = n$ versus $m < n$**
  - Holonomic versus nonholonomic

Types of Motion Planning Problems

- **Path planning versus motion planning**
  - Trajectory generation versus these lectures
- **Control inputs: $m = n$ versus $m < n$**
  - Holonomic versus nonholonomic
- **Online versus offline**
  - How reactive does your planner need to be?

Types of Motion Planning Problems

- **Path planning versus motion planning**
  - Trajectory generation versus these lectures
- **Control inputs: $m = n$ versus $m < n$**
  - Holonomic versus nonholonomic
- **Online versus offline**
  - How reactive does your planner need to be?
- **Optimal versus satisficing**
  - Minimum cost or just reach goal?
- **Exact versus approximate**
  - What is sufficiently close to goal?

Types of Motion Planning Problems

- **Path planning versus motion planning**
  - Trajectory generation versus these lectures
- **Control inputs: $m = n$ versus $m < n$**
  - Holonomic versus nonholonomic
- **Online versus offline**
  - How reactive does your planner need to be?
- **Optimal versus satisficing**
  - Minimum cost or just reach goal?
- **Exact versus approximate**
  - What is sufficiently close to goal?
- **With or without obstacles**
  - How challenging is the problem?

Properties of Motion Planners

- **Multiple-query versus single-query planning**
- **“Anytime” planning**
  - Continues to look for better solutions after first solution is found
- **Computational complexity**
  - Characterization of the amount of time a planner takes to run or the amount of memory it requires

Properties of Motion Planners

- **Multiple-query versus single-query planning**
- **“Anytime” planning**
  - Continues to look for better solutions after first solution is found
- **Computational complexity**
  - Characterization of the amount of time a planner takes to run or the amount of memory it requires
- **Completeness**
  - A planner is **complete** if it is guaranteed to find a solution in finite time if one exists, and report failure if no feasible plan exists
  - A planner is **resolution complete** if it is guaranteed to find a solution, if one exists, at the resolution of a discretized representation
  - A planner is **probabilistically complete** if the probability of finding a solution, if one exists, tends to 1 as planning time goes to infinity

Motion Planning Methods

- **Complete methods:** exact representations of the geometry of the problem and space
- **Grid methods:** discretize $\mathcal{C}_{\text{free}}$ and search the grid from q_{start} to goal
- **Sampling Methods:** randomly sample from the C-space, evaluate if the sample is in $\mathcal{X}_{\text{free}}$, and add new sample to previous samples
- **Virtual potential fields:** create forces on the robot that pull it toward goal and away from obstacles
- **Nonlinear optimization:** minimize some cost subject to constraints on the controls, obstacles, and goal
- **Smoothing:** given some guess or motion planning output, improve the smoothness while avoiding collisions

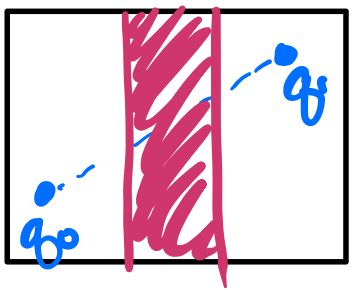
Configuration Space Obstacles

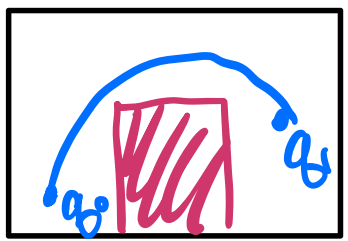
- We want to partition our C-space into parts:

$C = C_{free} \cup C_{obs}$

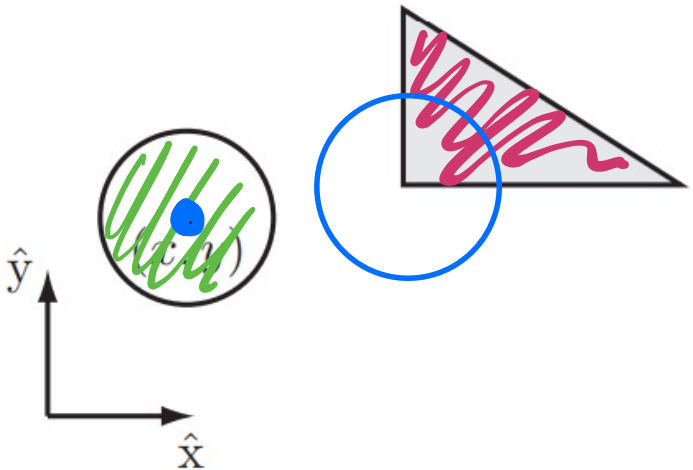
obstacles
joint limits

- If obstacles break C_{free} into separate components and q_0 and q_1 are not in the same connected components, then there is no collision free path

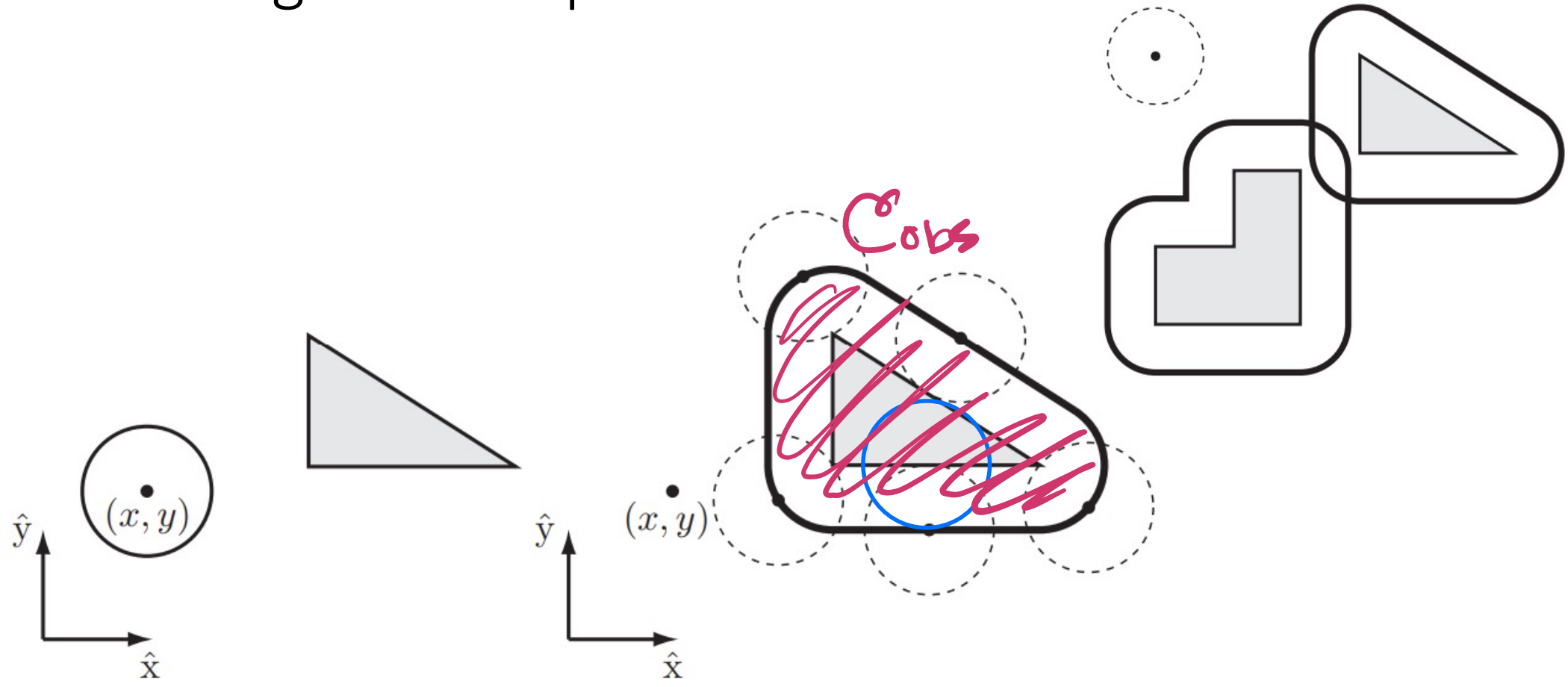




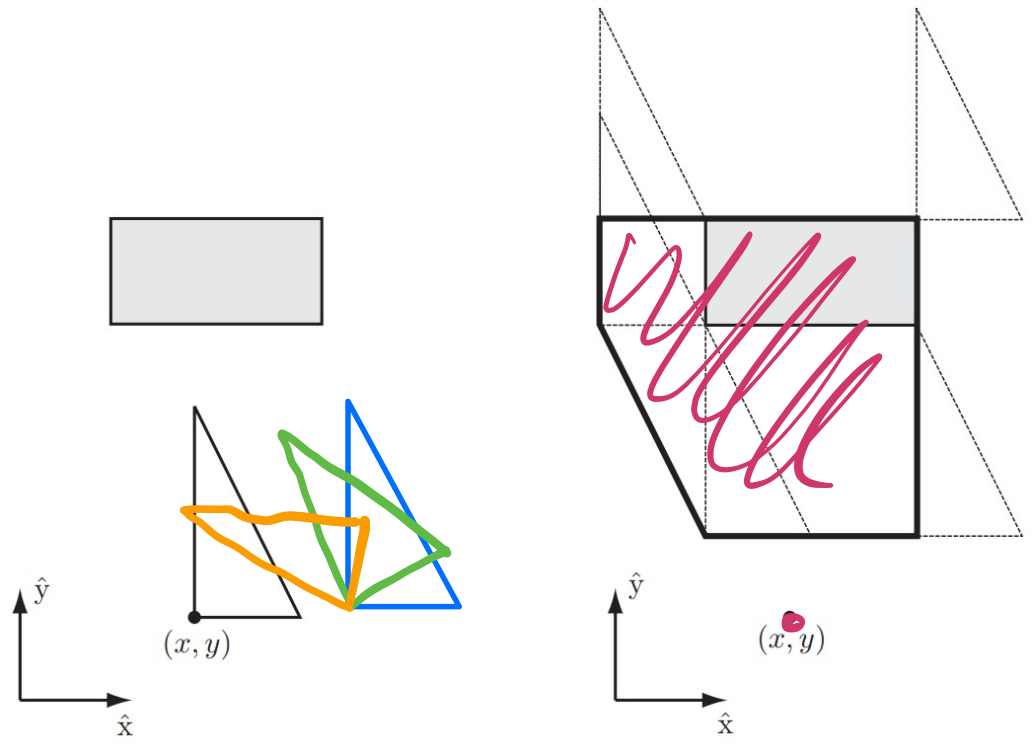
Configuration Space: Circular Mobile Bot



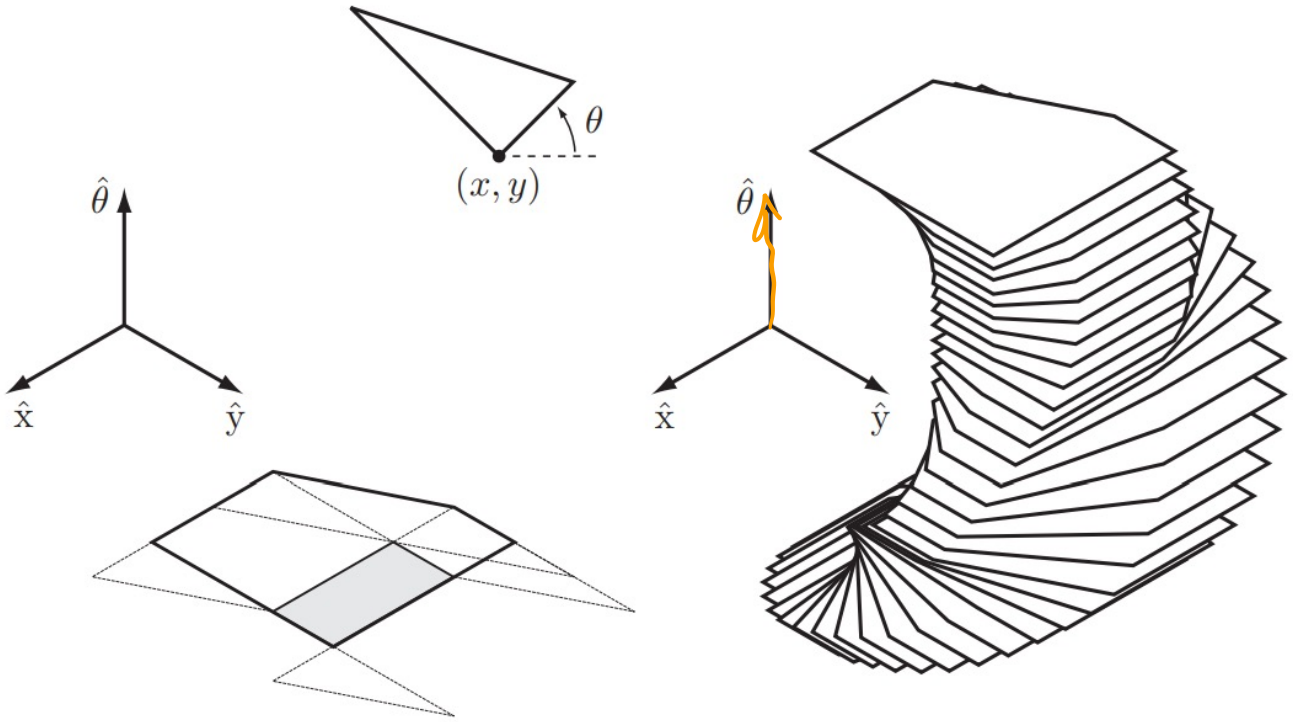
Configuration Space: Circular Mobile Bot



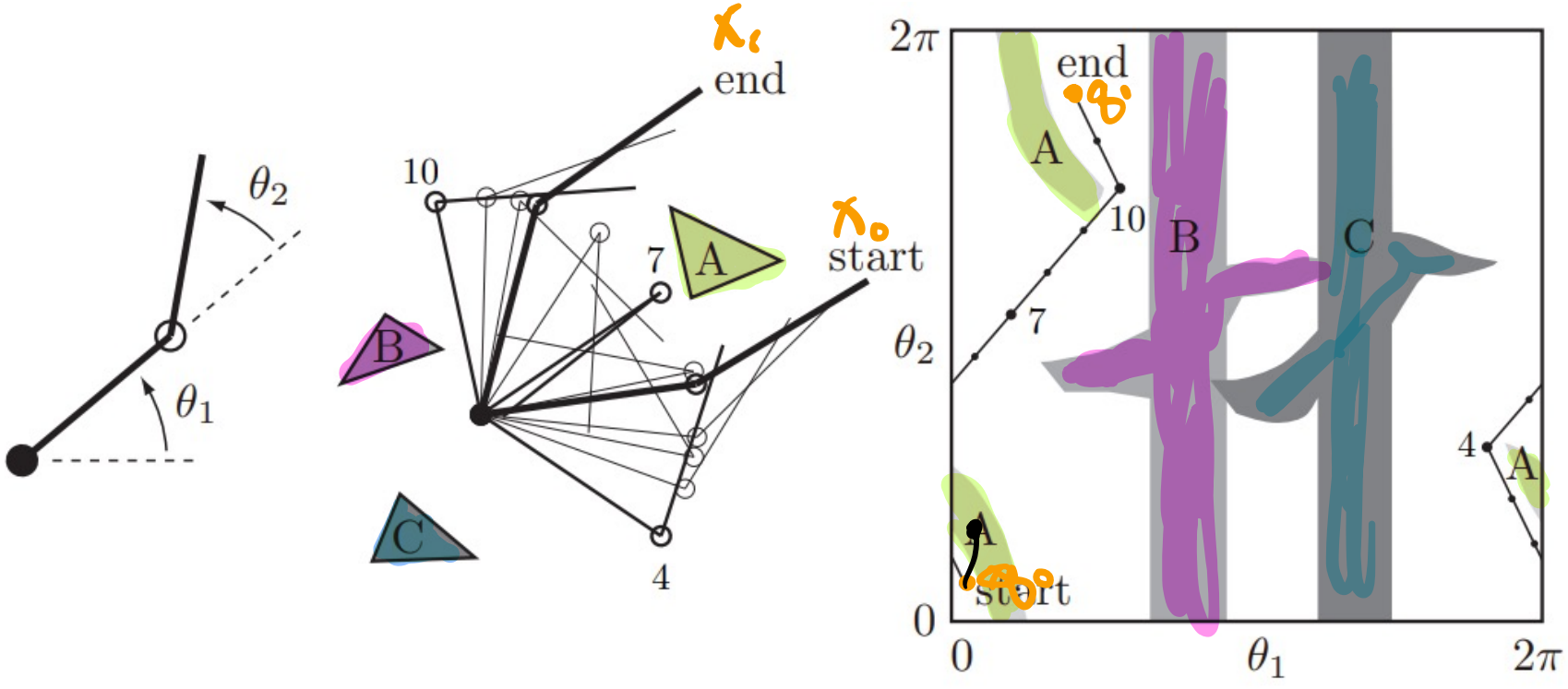
Configuration Space: bot that translates + ~~rotates~~



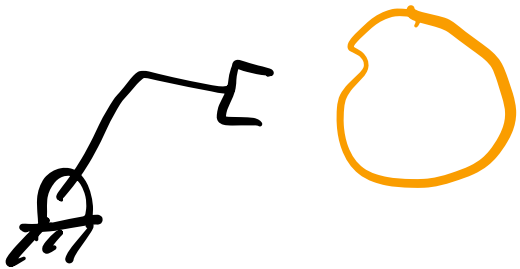
Configuration Space: bot that translates + rotates



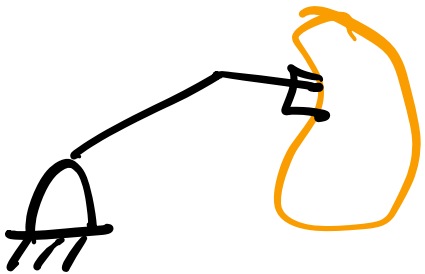
Configuration Space: 2R Planar Arm



Collision Detection



No contact



Contact

Penetration

We use a function $d(q, \mathcal{B})$ to measure the distance between the robot and the obstacle

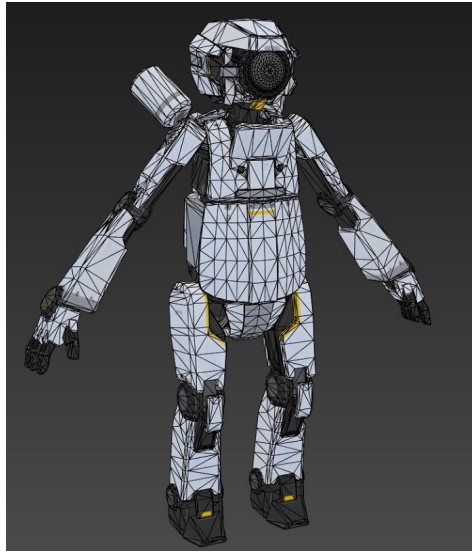
$d(q, \mathcal{B}) > 0$ -> No contact

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

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

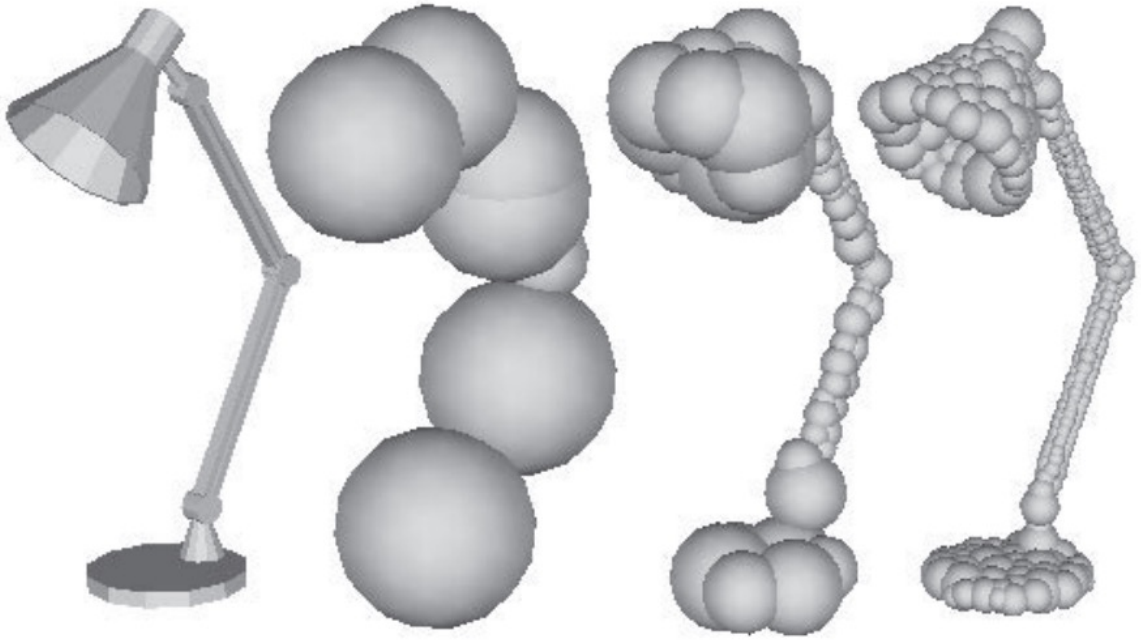
How to calculate the distance between robots and the obstacles?

- Naïve method: check every individual points on the robot and the obstacles
- Gilbert-Johnson-Keerthi (GJK) algorithm: computing the distance between con convex bodies that represented by triangular meshes



Spherical Approximations

One simple method is to approximate the robot and obstacles as unions of overlapping spheres



Distance Measures

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:

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

$$\forall\, i = 1, \ldots, k$$
$$j = 1, \ldots, l$$

Summary

- Defined and discussed concepts relating to **motion planning**
- Discussed configuration space components $\boldsymbol{\mathcal{C}_{\mathbf{free}}}$ and $\boldsymbol{\mathcal{C}_{obs}}$
- Gave example distance measurement approaches to check **collision detection**
- **Next time:** Learn about graphs, searching, and RRT