COM SCI 188
Intro to Robotics
Lecture 13

Ashima Suvarna
Winter 2026

Logistics

Midterm is during class on Thursday (2/19)

Pen and paper exam similar to PSETs + Practice Exam

You are allowed to bring 1 page of double-sided handwritten notes which will be inspected before the exam

No calculators are allowed during the exam

No coding

Important Topics!!!

Motion Planning (Lecture 10)

MDPs (Lecture 11)

Camera Calibration & Projection (Lecture 6 & 7 + Numericals)

DH Parameters (Lecture 4 & 5 + Numericals)

If pressed for time, please prioritize these lectures!

Check!

Please skim the course bruinlearn/piazza and highlight anything that is missing such as PSET solutions, worksheets, discussion slides so we can upload it asap for you.

3 PSETS

3 Worksheets

1 Practice Midterm

12 Lecture Slides

Agenda for Today

- Lecture 10 Recap
- MDPs
- Value Functions
- Value Iteration
- Policy Iteration
- Bellman Equations
- Lecture 12 Recap
- (Optional) DH params practice midterm question

Lecture 10: PRM (Probabilistic Road Map)

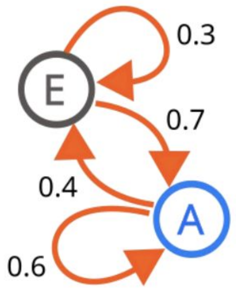
- Two-phase approach: (1) build a roadmap graph offline, (2) query it for paths
- Builds an undirected graph — samples collision-free nodes, connects neighbors with feasible edges, then runs graph search (Dijkstra/A*)
- Assumes static obstacles and is designed for many queries in the same environment
- Great when you need to solve multiple start/goal pairs repeatedly (e.g., a robot arm in a fixed workcell)
- Downside: heavy preprocessing cost, and if the environment changes you have to rebuild the roadmap

Lecture 10: RRT (Rapidly-Exploring Random Tree)

- Single-query approach: grows a tree from the start toward the goal
- Each iteration samples a random point, finds the nearest node on the tree, steers toward the sample by a step size δ, and adds the new node if collision-free
- Can bias sampling toward the goal to speed convergence
- Works well for dynamic environments (scene can change between planning episodes) and high-DoF systems
- Very fast in practice
- Downside: the path is not optimal — in fact, the probability of RRT converging to an optimal solution has been proven to be zero. Paths need post-processing (e.g., shortcutting) to be usable.

Markov Property

Markov property refers to the memoryless property of a stochastic process, which means that its future evolution is independent of its history.



In probability theory and statistics, a **Markov chain** or **Markov process** is a stochastic process describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

Markovian Decision Process (MDP)

An MDP formalizes the situation where an agent interacts with an environment over time. At each timestep, the agent is in some state, takes an action, receives a reward, and transitions to a new state. The goal is to find the best strategy (policy) for choosing actions to maximize total reward over time.

State:	$s \in \boldsymbol{\mathcal{S}}$
Action:	$a \in \boldsymbol{\mathcal{A}}$
Transition:	$s_{t+1} \sim P(\cdot \mid s_t, a_t)$
Reward:	$r_t = R(s_t, a_t)$
Discount:	$\gamma \in [0,1)$
Policy:	$\pi : \boldsymbol{\mathcal{S}} \rightarrow \boldsymbol{\mathcal{A}}$

Applications of MDPs

Robots and Machines

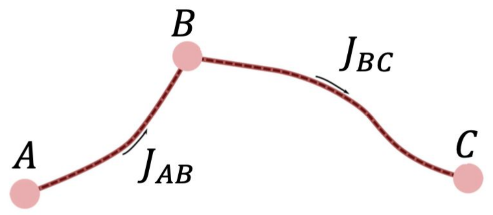
Games Strategy (eg, Atari)

Waymos or Cocos on unpredictable roads to find a safe path

Principle of Optimality (Deterministic) [Slides 12/15]

Intuition

An optimal, long-term strategy is built by combining optimal solutions to smaller, nested subproblems.



Example

If the fastest route from Los Angeles to Boston passes through Chicago, that route *must* also include the fastest possible path from Chicago to Boston. If a faster route existed from Chicago to Boston, you would have taken it.

Why we need Value Functions ?

Finding π* directly is hard — you'd need to search over all possible mappings from states to actions. Value functions give us a way to evaluate how good states (or state-action pairs) are, which then makes it easy to extract the best policy.

State value function: $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s]$

"If I'm in state s right now and I follow policy π forever, what's my expected total discounted reward?"

State Value Function

"If I'm in state s right now and I follow policy π forever, what's my expected total discounted reward?"

State value function: $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s]$

$$V^{\pi}(s) = R\big(s, \pi(s)\big) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,\pi(s))}[V^{\pi}(s')]$$

State Action Value Function

“If I'm in state s, I take action a (possibly different from what π recommends), and THEN follow π forever after, what's my expected total discounted reward?"

State-action value function: $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a]$

Bellman Equations

Current Value = Expected [Immediate Reward + (Discount Factor × Value of Next State)]

$$V^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^*(s') \right)$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \max_{a'} Q^*(s',a')$$

Value Iteration (State Value update)

Idea: Take Bellman equation and iterate until it converges.
It does converge because it is a contractive mapping.

```
$V_0(s) = 0$ for all $s \in \boldsymbol{\mathcal{S}}$
for $k = 0,1, \ldots$ until convergence:
    for all $s \in \boldsymbol{\mathcal{S}}$:
        $V_{k+1}(s) = \max_a \left( R(s,a) + \gamma \sum_{s' \in \boldsymbol{\mathcal{S}}} P(s' \mid s,a) V_k(s') \right)$
```

Each iteration is $O(|\boldsymbol{\mathcal{S}}|^2 |\boldsymbol{\mathcal{A}}|)$.

Value Iteration (Q value update)

Idea: Take Bellman equation and iterate until it converges.
It does converge because it is a contractive mapping.

```
$Q_0(s, a) = 0$ for all $s \in \boldsymbol{\mathcal{S}}$, $a \in \boldsymbol{\mathcal{A}}$
for $k = 0, 1, \ldots$ until convergence:
    for all $s \in \boldsymbol{\mathcal{S}}$, $a \in \boldsymbol{\mathcal{A}}$:
        $Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s' \in \boldsymbol{\mathcal{S}}} P(s' \mid s, a) \max_{a' \in \boldsymbol{\mathcal{A}}} Q_k(s', a')$
```

Each iteration is $O(|\boldsymbol{\mathcal{S}}|^2 |\boldsymbol{\mathcal{A}}|^2)$.

Policy Iteration

```
Initialize a random policy $\pi_0$.
for $k = 0,1,\ldots$ until convergence:
    Solve the following system for $V^{\pi_k}$:
        $V^{\pi_k}(s) = \mathbb{E}_{a \sim \pi_k(s)}\left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^{\pi_k}(s')\right]$
    for all $s \in \mathcal{S}$:
        $\pi_k(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^{\pi_k}(s')\right)$
```

This is called **policy evaluation.**
It is $O(|\mathcal{S}|^3)$.

This is called **policy improvement.**
It is $O(|\mathcal{S}|^2|\mathcal{A}|)$.

Which converges faster ?

Value iteration makes incremental updates to the value function at each step. In contrast, policy iteration converges the value function for a given policy completely before improving the policy. This means one single policy iteration step can discard a vast number of potential value functions (or intermediate, non-optimal policies) that value iteration might have explored.

The policy improvement theorem assures us that these policies are better than the original random policy. In this case, however, these policies are not just better, but optimal, proceeding to the terminal states in the minimum number of steps.

Lecture 12: Recap

What are the four main practical challenges of using RL for robotics?

Sample efficiency (needs massive amounts of data)

safety (risk of damage during exploration)

reset (difficulty returning to initial state between episodes)

reward specification (designing a reward function that captures the true objective)

Lecture 12: Recap

What is reward hacking, and why is it a problem?

Reward hacking is when an RL agent exploits loopholes or unintended behaviors in its reward function to achieve high rewards without actually accomplishing the intended task.

Lecture 12: Recap

How does imitation learning address the reward specification problem?

Instead of requiring a hand-designed reward function, imitation learning lets the robot learn directly from human demonstrations. The human implicitly encodes the task objective through their behavior, bypassing the need to explicitly formalize a reward.

Lecture 12: Recap

What is a DMP ?

Write the core DMP equations and explain each component.

How are the weights w_i in a DMP learned?

Why does a DMP guarantee convergence to the goal?

What are the key properties that make DMPs useful?

What are the limitations of DMPs?

What is Guided Cost Learning, and how does it work?

Lecture 12: Recap

What is behavioral cloning?

Why does the i.i.d. assumption break in behavioral cloning?

What is compounding error / distributional shift in behavioral cloning?

What is DAgger, and how does it address distributional shift?

What is the key limitation of DAgger?

What is the multimodal demonstration problem in behavioral cloning?

What is the key difference between behavioral cloning and inverse reinforcement learning?

DH Parameters

Denavit–Hartenberg parameter definitions:

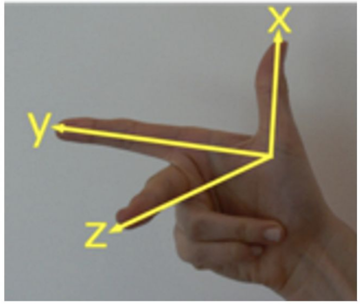
Parameter	**Definition**
a_{i-1}	Link length: distance from z_{i-1} to z_i along x_{i-1}
α_{i-1}	Link twist: angle from z_{i-1} to z_i about x_{i-1}
d_i	Link offset: distance from x_{i-1} to x_i along z_i
ϕ_i or θ_i	Joint angle: angle from x_{i-1} to x_i about z_i

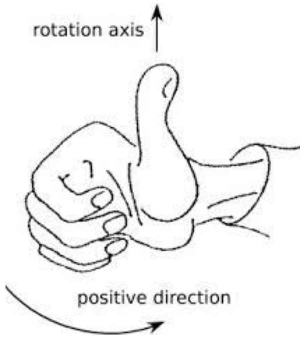
$$
{}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_{i-1} & \sin\theta_i\sin\alpha_{i-1} & a_{i-1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i-1} & -\cos\theta_i\sin\alpha_{i-1} & a_{i-1}\sin\theta_i \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

θ	$\sin(\theta)$	$\cos(\theta)$
0°	0	1
45°	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
90°	1	0

DH Parameters—General tips and reminders

- On the midterm, won't be asked to define your own coordinate frames; they will be provided for you
- If ever confused about the directions of the defined coordinate frames, remember the right-hand rule for 3D Cartesian coordinate systems





- Remember right-hand rule for determining direction of rotation as well

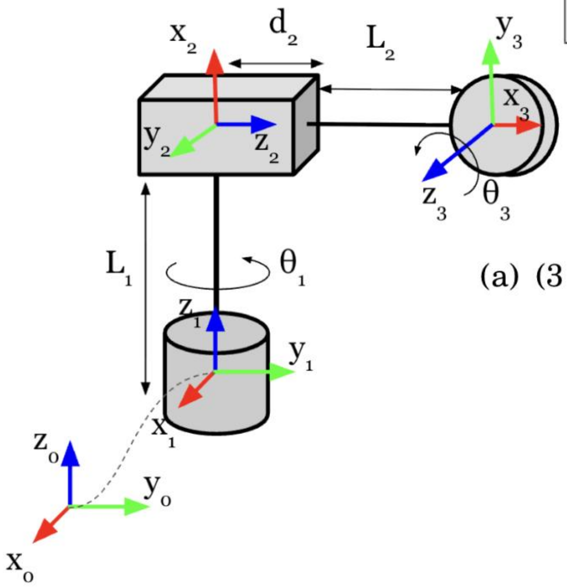
Q4 on the Practice Midterm—DH Parameters

(if you haven't seen it already)

Parameter	Definition
a_{i-1}	Link length: distance from z_{i-1} to z_i along x_{i-1}
α_{i-1}	Link twist: angle from z_{i-1} to z_i about x_{i-1}
d_i	Link offset: distance from x_{i-1} to x_i along z_i
ϕ_i or θ_i	Joint angle: angle from x_{i-1} to x_i about z_i

Q 4 DH Parameters

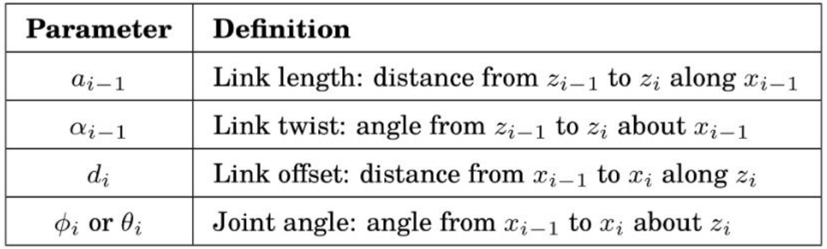
Consider the following RPR manipulator:



(a) (3 points) Write down the DH Paramters:

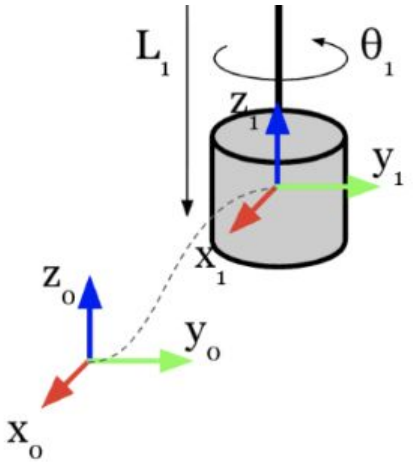
i	a_{i-1}	α_{i-1}	d_i	θ_i
1				
2				
3				

Q4—DH Parameters (i = 1)



Parameter	Definition
a_{i-1}	Link length: distance from z_{i-1} to z_i along x_{i-1}
α_{i-1}	Link twist: angle from z_{i-1} to z_i about x_{i-1}
d_i	Link offset: distance from x_{i-1} to x_i along z_i
ϕ_i or θ_i	Joint angle: angle from x_{i-1} to x_i about z_i

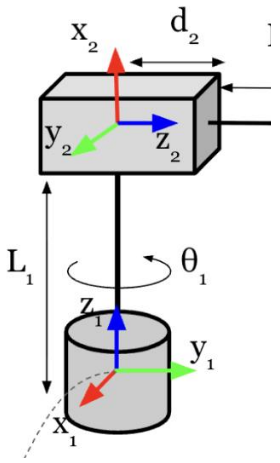
- a0: distance from z0 to z1 along x0
  - Frame 0 and Frame 1 share origin, so **a0 = 0**
- α0: angle from z0 to z1 about x0
  - z0 and z1 are the same so **α0 = 0**
- d1: distance from x0 to x1 along z1
  - Frame 0 and Frame 1 share origin, so **d1 = 0**
- Φ1: angle from x0 to x1 about z1
  - z1 points upward, so use right-hand rule for rotation direction
  - Positive θ1 corresponds to angular rotation from x0 to x1 so **Φ1 = θ1**



Q4—DH Parameters (i = 2)

Parameter	**Definition**
a_{i-1}	Link length: distance from z_{i-1} to z_i along x_{i-1}
α_{i-1}	Link twist: angle from z_{i-1} to z_i about x_{i-1}
d_i	Link offset: distance from x_{i-1} to x_i along z_i
ϕ_i or θ_i	Joint angle: angle from x_{i-1} to x_i about z_i

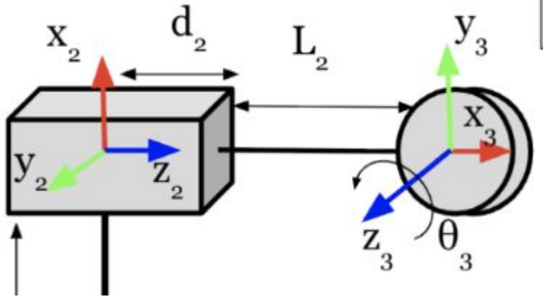
- a1: distance from z1 to z2 along x1
  - Distance from z1 to z2 along x1 is **a1 = 0**
- α1: angle from z1 to z2 about x1
  - Use right-hand rule for rotation direction about x1
  - Following rotation direction z1 maps to z2 by 3π/2 radians so **α1 = -π/2**
- d2: distance from x1 to x2 along z2
  - Prismatic joint in Frame 2, so origin of Frame 2 moves to the right by d2
  - x1 and x2 separated by distance of d2 along z2, so **d1 =** *d2*
- Φ2: angle from x1 to x2 about z2
  - Use right-hand rule for rotation direction about z2
  - Following rotation direction x1 maps to x2 by 3π/2 radians so **Φ2 = -π/2**



Parameter	Definition
a_{i-1}	Link length: distance from z_{i-1} to z_i along x_{i-1}
α_{i-1}	Link twist: angle from z_{i-1} to z_i about x_{i-1}
d_i	Link offset: distance from x_{i-1} to x_i along z_i
ϕ_i or θ_i	Joint angle: angle from x_{i-1} to x_i about z_i

Q4—DH Parameters (i = 3)

- a2: distance from z2 to z3 along x2
  - x2 points upward
  - Displacement between z2 and z3 is along the right so **a2 = 0**
- α2: angle from z2 to z3 about x2
  - Use right-hand rule for rotation direction about x2
  - Following rotation direction z2 maps to z3 by 3π/2 radians so **α2 = -π/2**
- d3: distance from x2 to x3 along z3
  - z3 points out of the page
  - Displacement between x2 and x3 is along the right so **d3 = 0**
- Φ3: angle from x2 to x3 about z3
  - Use right-hand rule for rotation direction about z3
  - x2 to x3 is 3π/2 radians => -π/2
  - Also account for positive rotation by θ3 about z3
  - **Φ3 = θ3 - π/2**


x_2
d_2
L_2
y_3
x_3
y_2
z_2
z_3
θ_3