ECE276B: Planning & Learning in Robotics
Lecture 1: Introduction

Instructor:
Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistant:
Hanwen Cao: h1cao@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

What is this class about?

- **ECE276A**: sensing and estimation in robotics:
  - how to model robot motion and observations
  - how to estimate (a distribution of) the robot/environment state $\mathbf{x}_t$ from the history of observations $\mathbf{z}_{0:t}$ and control inputs $\mathbf{u}_{0:t-1}$
- **ECE276B**: planning and decision making in robotics:
  - how to select control inputs $\mathbf{u}_{0:t-1}$ to accomplish a task
- **References** (optional):
  - Dynamic Programming and Optimal Control: Bertsekas
  - Planning Algorithms: LaValle (`http://planning.cs.uiuc.edu`)
  - Reinforcement Learning: Sutton & Barto (`http://incompleteideas.net/book/the-book.html`)
  - Calculus of Variations and Optimal Control Theory: Liberzon (`http://liberzon.csl.illinois.edu/teaching/cvoc.pdf`)

Logistics

- Course website: `https://natanaso.github.io/ece276b`
- Includes links to:
  - **Canvas**: lecture recordings
  - **Piazza**: course announcement, Q&A, discussion – <u>check Piazza regularly</u>
  - **Gradescope**: homework submission and grades
- Assignments:
  - 3 theoretical homeworks (16% of grade)
  - 3 programming assignments in **python** + project report:
    - Project 1: Dynamic Programming (18% of grade)
    - Project 2: Motion Planning (18% of grade)
    - Project 3: Optimal Control (18% of grade)
  - Final exam (30% of grade)
- Grading:
  - standard grade scale (93%+ = A) plus curve based on class performance (e.g., if the top students have grades in the 86% - 89% range, then this will correspond to letter grade A)
  - **no late submissions**: work submitted past the deadline receives 0 credit

Prerequisites

- **Probability theory**: random variables, probability density functions, expectation, covariance, total probability, conditioning, Bayes rule
- **Linear algebra/systems**: eigenvalues, symmetric positive definite matrices, linear equations, linear systems of ODEs, matrix exponential
- **Optimization**: unconstrained optimization, gradient descent
- **Programming**: experience with at least one language (python/C++/Matlab), classes/objects, data structures (e.g., queue, list), data input/output processing, plotting
- It is up to you to judge if you are ready for this course!
  - Consult with your classmates who took ECE276A
  - Take a look at the material from last year: `https://natanaso.github.io/ece276b2021`
  - If the first assignment seems hard, the rest will be hard as well

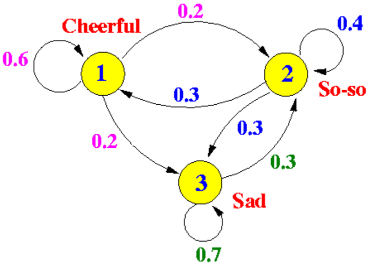
Syllabus (Tentative)

Date	Lecture	Materials	Assignments
Mar 29	Introduction		
Mar 31	Markov Chains	Grinstead-Snell-Ch11	
Apr 05	Markov Decision Processes	Bertsekas 1.1-1.2	
Apr 07	Dynamic Programming	Bertsekas 1.3-1.4	HW1, PR1
Apr 12	Deterministic Shortest Path	Bertsekas 2.1-2.3	HW1 Solutions
Apr 14	Configuration Space	LaValle 4.3, 6.2-6.3	
Apr 19	Search-based Planning	LaValle 2.1-2.3, JPS	
Apr 21	Catch-up		
Apr 26	Anytime Incremental Search	RTAA*, ARA*, AD*, Anytime Search	HW2, PR2
Apr 28	Sampling-based Planning	LaValle 5.5-5.6	HW2 Solutions
May 03	Stochastic Shortest Path	Bertsekas 7.1-7.3	
May 05	Bellman Equations I	Sutton-Barto 4.1-4.4	
May 10	Bellman Equations II	Sutton-Barto 4.5-4.8	
May 12	Catch-up		
May 17	Model-free Prediction	Sutton-Barto 6.1-6.3	HW3, PR3
May 19	Model-free Control	Sutton-Barto 6.4-6.7	HW3 Solutions
May 24	Value Function Approximation	Sutton-Barto Ch.9	
May 26	Continuous-time Optimal Control	Bertsekas 3.1-3.2	
May 31	Pontryagin's Minimum Principle	Bertsekas 3.3-3.4, Liberzon Ch. 2.4 and Ch. 4	
Jun 02	Linear Quadratic Control	Bertsekas 4.1	
Jun 09	Final Exam		

- Check website for updates: `https://natanaso.github.io/ece276b`

Markov Chain and Markov Decision Process

- **Markov Chain** (MC): a probabilistic model used to represent the state evolution of a system
  - The state $\mathbf{x}_t$ can be discrete or continuous and is fully observed
  - The state transitions are random and uncontrolled, determined by a transition matrix or function



- **Markov Decision Process** (MDP): a Markov chain whose transitions are controlled by system control inputs $\mathbf{u}_t$
- Motion planning, optimal control, and reinforcement learning problems are defined using a Markov decision process

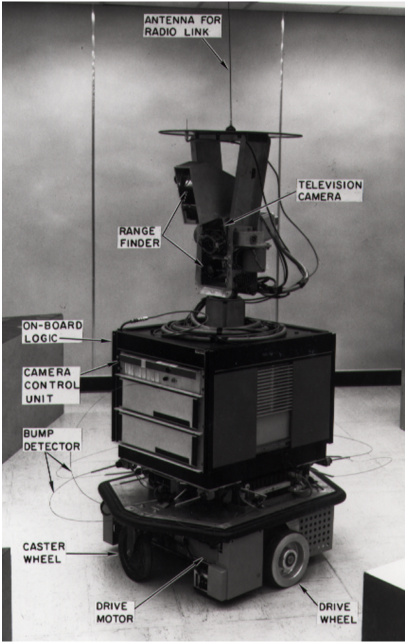
$$P = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.3 & 0.4 & 0.3 \\ 0.0 & 0.3 & 0.7 \end{bmatrix}$$

$$P_{ij} = \mathbb{P}(x_{t+1} = j \mid x_t = i)$$

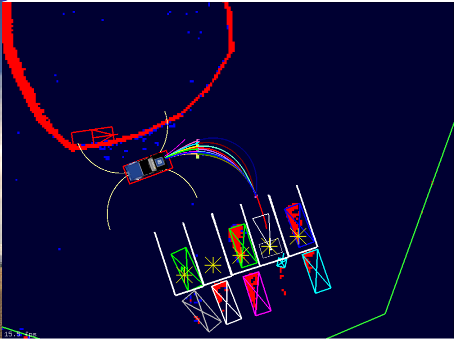
Motion Planning

A* Search

- Invented by Hart, Nilsson and Raphael of Stanford Research Institute in 1968 for the Shakey robot
- MDP with deterministic transitions, i.e., directed graph
- Minimize cumulative transition costs subject to a goal constraint
- Graph search using a specific node visitation rule
- Video: `https://youtu.be/qXdn6ynwpiI?t=3m55s`

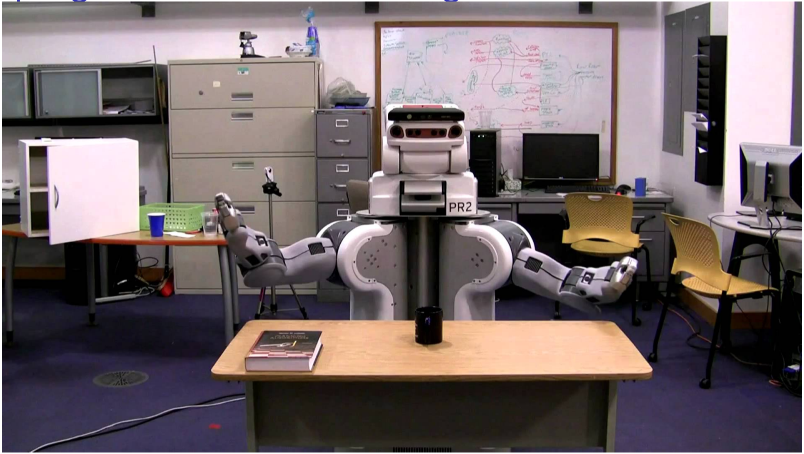


Search-based Motion Planning



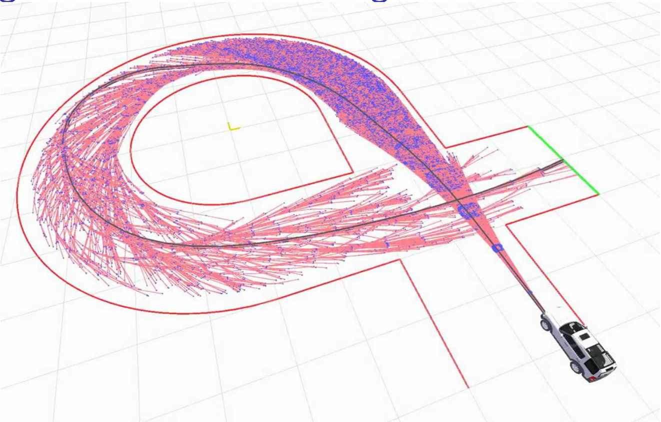
- CMU's autonomous car used search-based motion planning in the DARPA Urban Challenge in 2007
- Likhachev and Ferguson, "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles," IJRR'09
- Video: `https://www.youtube.com/watch?v=4hFhl0Oi8KI`
- Video: `https://www.youtube.com/watch?v=qXZt-B7iUyw`
- Paper: `http://journals.sagepub.com/doi/pdf/10.1177/0278364909340445`

Sampling-based Motion Planning



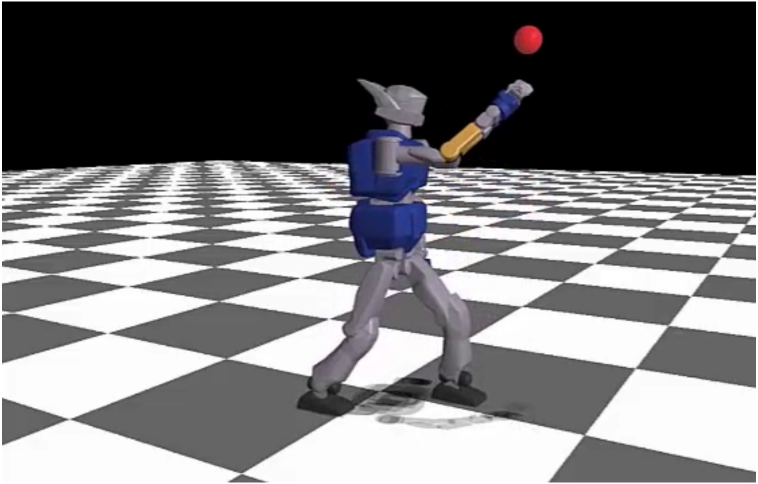
- RRT algorithm on the PR2 – planning with both arms (12 DOF)
- Karaman and Frazzoli, “Sampling-based algorithms for optimal motion planning,” IJRR’11
- Video: `https://www.youtube.com/watch?v=vW74bC-Ygb4`
- Paper: `http://journals.sagepub.com/doi/pdf/10.1177/0278364911406761`

Sampling-based Motion Planning



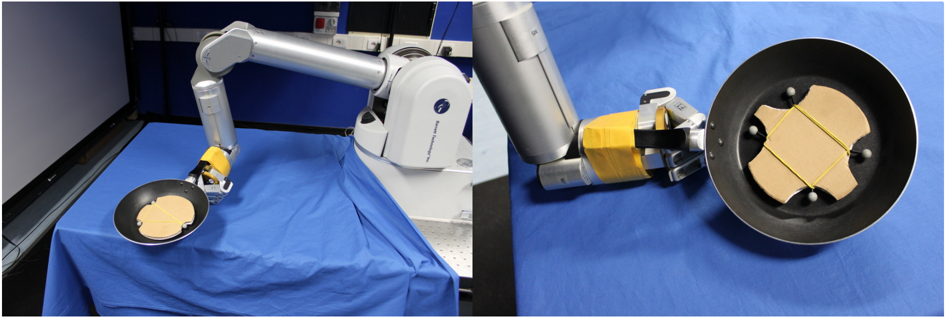
- RRT* algorithm on a high-fidelity car model – 270 degree turn
- Karaman and Frazzoli, "Sampling-based algorithms for optimal motion planning," IJRR'11
- Video: `https://www.youtube.com/watch?v=p3nZHnOWhrg`
- Video: `https://www.youtube.com/watch?v=LKL5qRBiJaM`
- Paper: `http://journals.sagepub.com/doi/pdf/10.1177/0278364911406761`

Optimal Control using Dynamic Programming



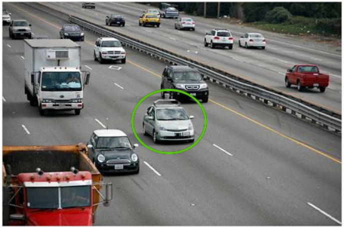
- Tassa, Mansard and Todorov, "Control-limited Differential Dynamic Programming," ICRA'14
- Video: `https://www.youtube.com/watch?v=tCQSSkBH2NI`
- Paper: `http://ieeexplore.ieee.org/document/6907001/`

Model-free Reinforcement Learning



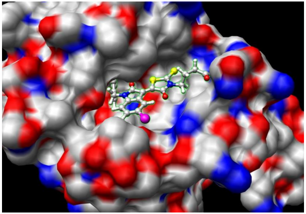
- A robot learns to flip pancakes
- Kormushev, Calinon and Caldwell, “Robot Motor Skill Coordination with EM-based Reinforcement Learning,” IROS’10
- Video: `https://www.youtube.com/watch?v=W_gxLKSsSIE`
- Paper: `http://www.dx.doi.org/10.1109/IROS.2010.5649089`

Applications of Optimal Control & Reinforcement Learning



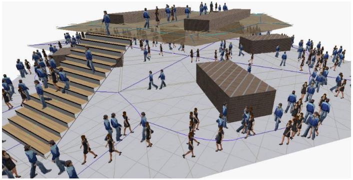
(a) Autonomous Driving

(b) Marketing

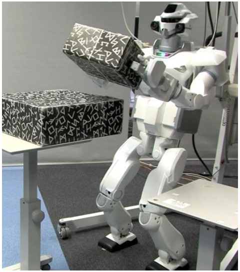


(c) Computational Biology

(d) Games



(e) Character Animation



(f) Robotics

Model

- discrete **time** $t \in \{0, \ldots, T\}$ with finite or infinite horizon T
- **state** $\mathbf{x}_t \in \mathcal{X}$ and **state space** $\mathcal{X}$
- **control** $\mathbf{u}_t \in \mathcal{U}$ and **control space** $\mathcal{U}$
- **noise** $\mathbf{w}_t$: random vector with known probability density function (pdf), independent of $\mathbf{w}_\tau$ for $\tau \neq t$ conditioned on $\mathbf{x}_t$ and $\mathbf{u}_t$
- **motion model**: a function f or equivalently a pdf p_f describing the change in the state $\mathbf{x}_t$ when a control input $\mathbf{u}_t$ is applied:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \qquad \text{or} \qquad \mathbf{x}_{t+1} \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$$

  - **Markov assumption**: $\mathbf{x}_{t+1}$ depends only on $\mathbf{u}_t$ and $\mathbf{x}_t$
- **stage cost** $\ell(\mathbf{x}, \mathbf{u})$ measures the cost of applying control $\mathbf{u}$ in state $\mathbf{x}$
- **terminal cost** $\mathfrak{q}(\mathbf{x})$ measures the cost of terminating at state $\mathbf{x}$

Problem Statement

- **control policy** $\pi_t : \mathcal{X} \mapsto \mathcal{U}$ maps a state $\mathbf{x}$ at time t to a control input $\mathbf{u}$
- A control policy π induces a system transition from state $\mathbf{x}_t$ at time t with control input $\mathbf{u}_t = \pi_t(\mathbf{x}_t)$ to state $\mathbf{x}_{t+1} \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$
- **value function** $V_t^\pi(\mathbf{x})$ of policy π is the expected long-term cost of starting at state $\mathbf{x}$ at time t and following transitions induced by π:

$$V_t^\pi(\mathbf{x}) := \mathbb{E}_{\mathbf{x}_{t+1:T}}\left[\underbrace{\mathfrak{q}(\mathbf{x}_T)}_{\text{terminal cost}} + \sum_{\tau=t}^{T-1} \underbrace{\ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau))}_{\text{stage cost}} \;\middle|\; \mathbf{x}_t = \mathbf{x}\right]$$

- **optimal control problem**: given initial state $\mathbf{x}$ at time t, determine a policy that minimizes the value function $V_t^\pi(\mathbf{x})$:
  - **optimal value**: $V_t^*(\mathbf{x}) = \min_\pi V_t^\pi(\mathbf{x})$
  - **optimal policy**: $\pi^*(\mathbf{x}) \in \arg\min_\pi V_t^\pi(\mathbf{x})$

Naming Conventions

- The problem is called:
  - **Motion planning** (MP): when the motion model p_f is known and deterministic and the cost functions ℓ, $\mathfrak{q}$ are known
  - **Optimal control** (OC): when the motion model p_f is known but may be stochastic and cost functions ℓ, $\mathfrak{q}$ are known
  - **Reinforcement Learning** (RL): when the motion model p_f and cost functions ℓ, $\mathfrak{q}$ are unknown but samples $\mathbf{x}_t$, $\ell(\mathbf{x}_t, \mathbf{u}_t)$, $\mathfrak{q}(\mathbf{x}_t)$ can be obtained from them
- Naming conventions differ:
  - **OC**: minimization, cost, state $\mathbf{x}$, control $\mathbf{u}$, policy μ
  - **RL**: maximization, reward, state $\mathbf{s}$, action $\mathbf{a}$, policy π
  - **ECE276B**: minimization, cost, state $\mathbf{x}$, control $\mathbf{u}$, policy π

Policy Types

- Controls may have long-term consequences, e.g., delayed cost/reward
- It may be better to sacrifice immediate rewards to gain long-term rewards:
  - A financial investment may take months to mature
  - Re-fueling a helicopter now might prevent a crash in several hours
  - Blocking an opponent move now might help winning chances many moves from now
- A policy defines fully at any time t and any state $\mathbf{x}$ which control $\mathbf{u}$ to apply
- A policy can be:
  - **stationary** $(\pi_0 \equiv \pi_1 \equiv \cdots) \subset$ **non-stationary** $(\pi_0 \not\equiv \pi_1 \not\equiv \cdots)$
  - **deterministic** $(\mathbf{u}_t = \pi_t(\mathbf{x}_t)) \subset$ **stochastic** $(\mathbf{u}_t \sim \pi_t(\cdot \mid \mathbf{x}_t))$
  - **open-loop** (sequence $\mathbf{u}_{0:T-1}$ regardless of $\mathbf{x}_t$) $\subset$ **closed-loop** ($\mathbf{u}_t = \pi_t(\mathbf{x}_t)$ depends on $\mathbf{x}_t$)

Problem Types

- **deterministic** (no motion noise) vs **stochastic** (with motion noise)
- **fully observable** ($\mathbf{z}_t = \mathbf{x}_t$) vs **partially observable** ($\mathbf{z}_t \sim p_h(\cdot|\mathbf{x}_t)$)
  - Markov Decision Process (MDP) vs Partially Observable Markov Decision Process (POMDP)
- **stationary** vs **non–stationary** (time-dependent motion $p_{f,t}$ and cost ℓ_t)
- **discrete** vs **continuous** state space $\mathcal{X}$
  - tabular approach vs function approximation
- **discrete** vs **continuous** control space $\mathcal{U}$:
  - tabular approach vs optimization
- **discrete** vs **continuous** time t
- **finite** vs **infinite** horizon T
- reinforcement learning (p_f, ℓ, $\mathfrak{q}$ are unknown):
  - **Model-based RL**: explicitly approximate the models $\hat{p}_f$, $\hat{\ell}$, $\hat{\mathfrak{q}}$ from data and apply optimal control algorithms
  - **Model-free RL**: directly approximate V_t^* and π_t^* without approximating the motion or cost models

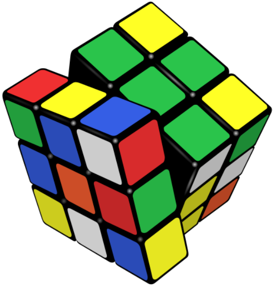
Example: Inventory Control

- Consider keeping an item stocked in a warehouse:
  - If there is too little, we may run out (not preferred).
  - If there is too much, the storage cost will be high (not preferred).

- Model:
  - $x_t \in \mathbb{R}$: stock available in the warehouse at the beginning of the t-th time period
  - $u_t \in \mathbb{R}_{\geq 0}$: stock ordered and immediately delivered at the beginning of the t-th time period (supply)
  - w_t: random demand during the t-th time period with known pdf. Note that excess demand is back-logged, i.e., corresponds to negative stock x_t
  - **Motion model**: $x_{t+1} = f(x_t, u_t, w_t) := x_t + u_t - w_t$
  - **Cost function**: $\mathbb{E}\left[\mathfrak{q}(x_T) + \sum_{t=0}^{T-1}\left(r(x_t) + cu_t - pw_t\right)\right]$ where
    - pw_t: revenue
    - cu_t: cost of items
    - $r(x_t)$: penalizes too much stock or negative stock
    - $\mathfrak{q}(x_T)$: remaining items we cannot sell or demand that we cannot meet

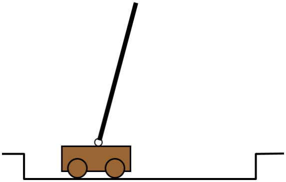
Example: Rubik's Cube

- Invented in 1974 by Ernõ Rubik
- Model:
  - State space size: $\sim 4.33 \times 10^{19}$
  - Control space size: 12
  - Cost: 1 for each time step
  - Deterministic, Fully Observable
- The cube can be solved in 20 or fewer moves



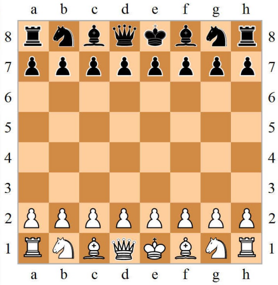
Example: Cart-Pole Problem

- Move a cart left, right to keep a pole balanced
- Model:
  - State space: 4-D continuous $(x, \dot{x}, \theta, \dot{\theta})$
  - Control space: $\{-N, N\}$
  - Cost:
    - 0 when in the goal region
    - 1 when outside the goal region
    - 100 when outside the feasible region
  - Deterministic, Fully Observable



Example: Chess

- Model:
  - State space size: $\sim 10^{47}$
  - Control space size: from 0 to 218
  - Cost: 0 each step, $\{-1, 0, 1\}$ at the end of the game
  - Deterministic, opponent-dependent state transitions (can be modeled as a game)
- The size of the game tree (all possible policies) is 10^{123}



Example: Grid World Navigation

- Navigate to a goal without crashing into obstacles
- Model:
  - State space: 2-D robot position
  - Control space: $\mathcal{U} = \{\mathit{left}, \mathit{right}, \mathit{up}, \mathit{down}\}$
  - Cost: 1 until the goal is reached, ∞ if an obstacles is hit
  - Can be deterministic or stochastic; fully or partially observable