ECE276B: Planning & Learning in Robotics
Lecture 0: Introduction

Instructor:
Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:
Zhichao Li: zhl355@eng.ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

What is this class about?

- **ECE276A**: sensing and state estimation in robotics:
  - how to model a robot's motion and observations
  - how to estimate (a distribution of) the robot 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** (not required):
  - 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 (sign up!):
  - **Canvas**: Zoom meeting schedule and lecture recordings
  - **Piazza**: discussion – it is your responsibility to check Piazza regularly because class announcements, updates, etc., will be posted there
  - **Gradescope**: homework submission and grades
- Assignments:
  - 4 theoretical homework sets (5% of grade each)
  - 4 programming assignments in **python** + project report:
    - Project 1: Dynamic Programming (20% of grade)
    - Project 2: Motion Planning (20% of grade)
    - Project 3: Optimal Control (20% of grade)
    - Project 4: Final project (20% of grade)
- Final project proposals due **May 14, 2020**
  - Selected by you. Teams of 2 allowed but not required.
- Grades:
  - assigned based on the class performance, i.e., there will be a curve
  - **no late policy**: work submitted past the deadline will receive 0 credit

Prerequisites

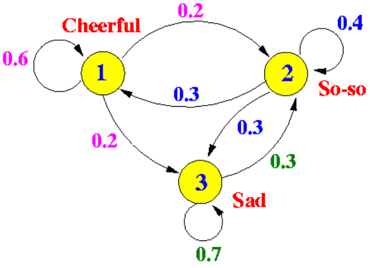
- **Probability theory**: random vectors, probability density functions, expectation, covariance, total probability, conditioning, Bayes rule
- **Linear algebra/systems**: eigenvalues, positive definiteness, linear systems of ODEs, matrix exponential
- **Optimization**: gradient descent
- **Programming**: experience with at least one language (python/C++/Matlab), classes/objects, data structures (e.g., queue, list), data input/output, 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/ece276b2019`
  - If the first assignment seems hard, the rest will be hard as well

Syllabus (Winter 2018)

Date	Lecture	Materials	Assignments
Mar 31	Markov Chains	Grinstead-Snell-Ch11	
Apr 02	Markov Decision Processes	Bertsekas 1.1-1.2	
Apr 07	Dynamic Programming	Bertsekas 1.3-1.4	HW1, PR1
Apr 09	Deterministic Shortest Path	Bertsekas 2.1-2.3	
Apr 14	Configuration Space	LaValle 4.3, 6.2-6.3	
Apr 16	Search-based Planning	LaValle 2.1-2.3, JPS	
Apr 21	Catch-up		HW2, PR2
Apr 23	Anytime Incremental Search	RTAA*, ARA*, AD*, Journal Paper	
Apr 28	Sampling-based Planning	LaValle 5.5-5.6	
Apr 30	Stochastic Shortest Path	Bertsekas 7.1-7.3	
May 05	Bellman Equations I	Sutton-Barto 4.1-4.4	
May 07	Bellman Equations II	Sutton-Barto 4.5-4.8	HW3, PR3
May 12	Catch-up		
May 14	Model-free Prediction	Sutton-Barto 6.1-6.3	
May 19	Model-free Control	Sutton-Barto 6.4-6.7	
May 21	Value Function Approximation	Sutton-Barto Ch.9	PR4
May 26	Continuous-time Optimal Control	Bertsekas 3.1-3.2	HW4
May 28	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 04	TBD		

Markov Chain

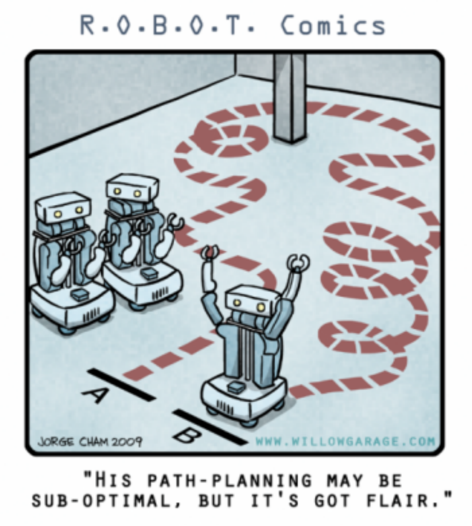
- A **Markov Chain** is a probabilistic model used to represent the evolution of a robot system
- The state $x_t \in \{1, 2, 3\}$ is fully observed (unlike HMM and Bayes filtering settings)
- The transitions are random, determined by a motion model but uncontrolled (just like in the HMM and Bayes filtering settings, the control input is known)
- A **Markov Decision Process** (MDP) is a Markov chain, whose transitions are controlled



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

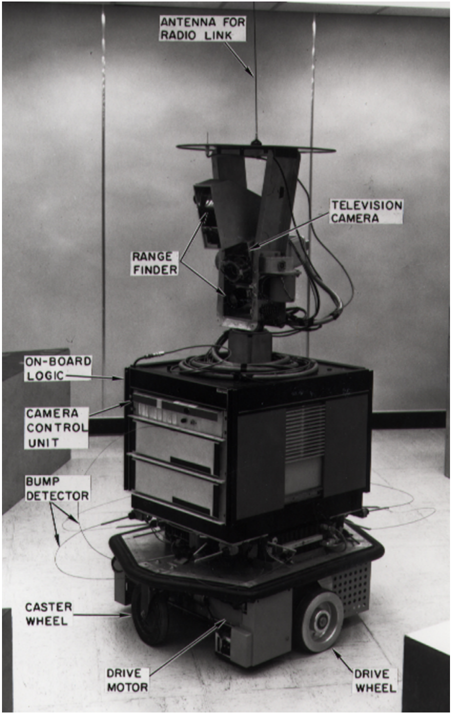
$$P_{ij} = \mathbb{P}(x_{t+1} = j \mid x_t = i)$$

Motion Planning

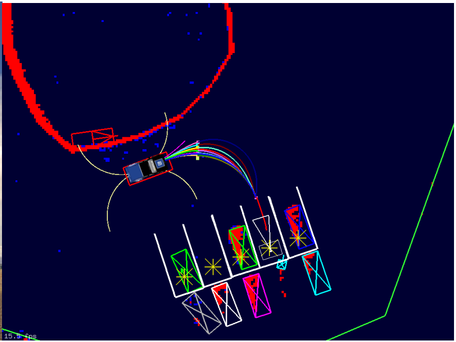


A* Search

- Invented by Hart, Nilsson and Raphael of Stanford Research Institute in 1968 for the Shakey robot
- Video: `https://youtu.be/qXdn6ynwpiI?t=3m55s`

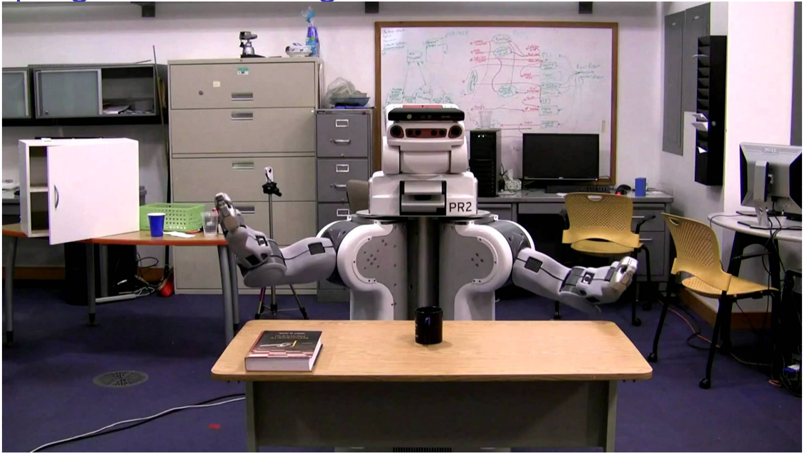


Search-based Planning



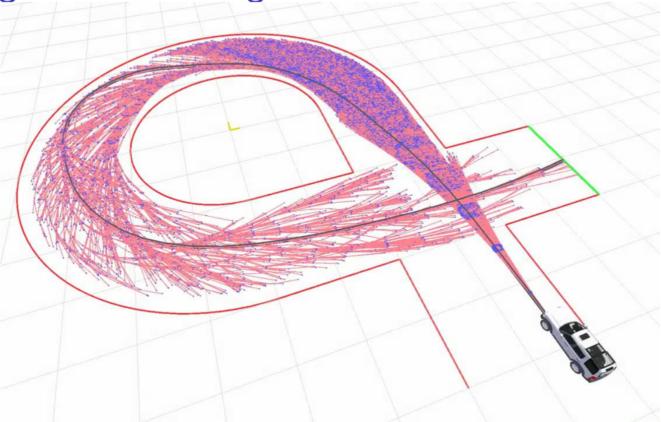
- CMU's autonomous car used search-based planning in the DARPA Urban Challenge in 2007
- Likhachev and Ferguson, "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles," IJRR'09
- Video: `https://www.youtube.com/watch?v=4hFhl0Oi8KI`
- Video: `https://www.youtube.com/watch?v=qXZt-B7iUyw`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

Sampling-based Planning



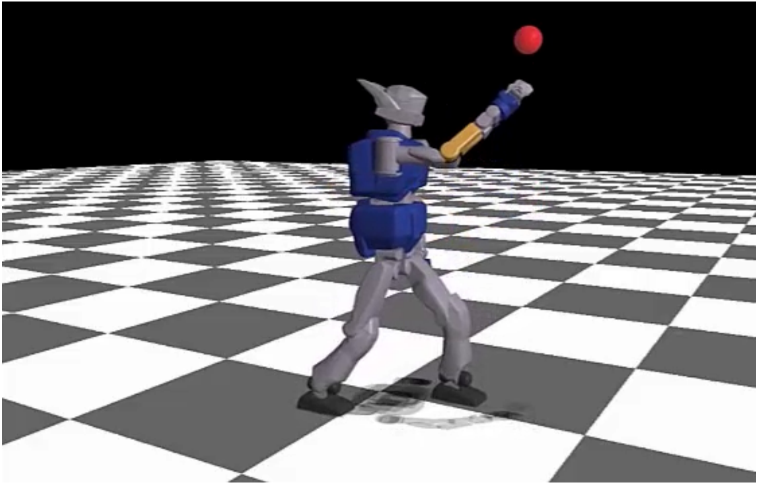
- RRT algorithm on the PR2 – planning with both arms (12 DOF)
- Karaman and Frazzoli, “Sampling-based algorithms for optimal motion planning,” IJRR’11
- Video: `https://www.youtube.com/watch?v=vW74bC-Ygb4`
- Paper: `http://journals.sagepub.com/doi/pdf/10.1177/0278364911406761`

Sampling-based Planning



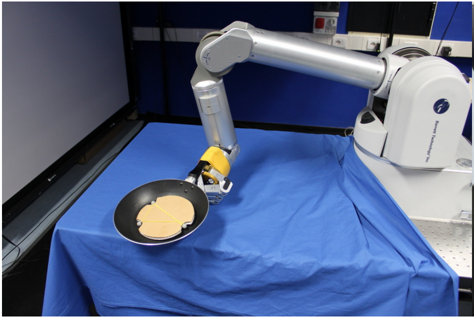
- RRT* algorithm on a race car – 270 degree turn
- Karaman and Frazzoli, "Sampling-based algorithms for optimal motion planning," IJRR'11
- Video: `https://www.youtube.com/watch?v=p3nZHnOWhrg`
- Video: `https://www.youtube.com/watch?v=LKL5qRBiJaM`
- Paper: `http://journals.sagepub.com/doi/pdf/10.1177/0278364911406761`

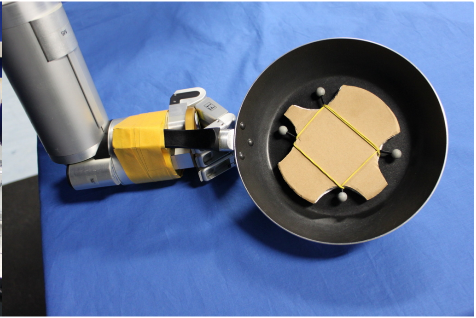
Dynamic Programming and Optimal Control



- 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

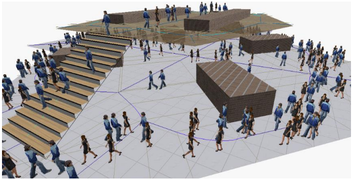




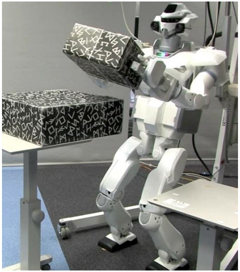
- Robot learns to flip pancakes
- Kormushev, Calinon and Caldwell, "Robot Motor Skill Coordination with EM-based Reinforcement Learning," IROS'10
- Video: `https://www.youtube.com/watch?v=W_gxLKSsSIE`
- Paper: `http://www.dx.doi.org/10.1109/IROS.2010.5649089`

Applications of Optimal Control & Reinforcement Learning

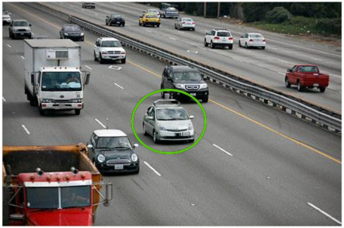
(a) Games



(b) Character Animation

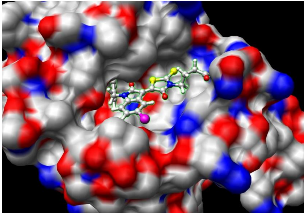


(c) Robotics



(d) Autonomous Driving

(e) Marketing



(f) Computational Biology

Problem Formulation

- **Motion model**: specifies how a dynamical system evolves

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t), \quad t = 0, \ldots, T-1$$

  - discrete time $t \in \{0, \ldots, T\}$
  - state $\mathbf{x}_t \in \mathcal{X}$
  - control $\mathbf{u}_t \in \mathcal{U}(\mathbf{x}_t)$ and $\mathcal{U} := \bigcup_{\mathbf{x} \in \mathcal{X}} \mathcal{U}(\mathbf{x})$
  - motion noise $\mathbf{w}_t$ (random vector) with known probability density function (pdf) and assumed conditionally independent of other disturbances $\mathbf{w}_\tau$ for $\tau \neq t$ for given $\mathbf{x}_t$ and $\mathbf{u}_t$
  - the motion model is specified by the nonlinear function f or equivalently by the pdf p_f of $\mathbf{x}_{t+1}$ conditioned on $\mathbf{x}_t$ and $\mathbf{u}_t$

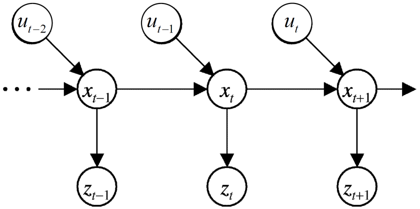
- **Observation model**: the state $\mathbf{x}_t$ might not be observable but perceived through measurements:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{v}_t) \sim p_h(\cdot \mid \mathbf{x}_t), \quad t = 0, \ldots, T$$

  - measurement noise $\mathbf{v}_t$ (random vector) with known pdf and conditionally independent of other disturbances $\mathbf{v}_\tau$ for $\tau \neq t$ and $\mathbf{w}_t$ for all t for given $\mathbf{x}_t$
  - the observation model is specified by the nonlinear function h or equivalently by the pdf p_h of $\mathbf{z}_t$ conditioned on $\mathbf{x}_t$

Problem Formulation

- **Markov Assumptions**
  - The state $\mathbf{x}_{t+1}$ only depends on the previous input $\mathbf{u}_t$ and state $\mathbf{x}_t$
  - The observation $\mathbf{z}_t$ only depends on the state $\mathbf{x}_t$



- **Problem structure**: due to the Markov assumptions, the joint distribution of the robot states $\mathbf{x}_{0:T}$, observations $\mathbf{z}_{0:T}$, and controls $\mathbf{u}_{0:T-1}$ satisfies:

$$p(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) =$$

$$\underbrace{p_0(\mathbf{x}_0)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}} \underbrace{\prod_{t=0}^{T-1} p(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{control policy}}$$

The Problem of Acting Optimally

- In general, states $\mathbf{x}_t$ are **partially observable** through the observations $\mathbf{z}_t$ based on the observation model p_h and the prior $p_0(\mathbf{x}_0)$
- A partially obserable problem can always be converted to a fully observed one by changing the state from $\mathbf{x}_t$ to the probability density function $p_{t|t}(\mathbf{x}_t) := p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$
- Thus, without loss of generality, we consider fully observable problems
- **Problem Statement**: given a motion model p_f and direct observations of the system state $\mathbf{x}_t$, determine control inputs $\mathbf{u}_{0:T-1}$ to minimize (maximize) a long-term cost (reward) function:

$$V_0^{\mathbf{u}_{0:T-1}}(\mathbf{x}_0) := \mathbb{E}_{\mathbf{x}_{1:T}}\left[\underbrace{\mathfrak{q}(\mathbf{x}_T)}_{\text{terminal cost}} + \sum_{t=0}^{T-1} \underbrace{\ell(\mathbf{x}_t, \mathbf{u}_t)}_{\text{stage cost}} \,\middle|\, \mathbf{x}_0, \mathbf{u}_{0:T-1}\right]$$

Problem Solution: Control Policy

- The solution to an OC/RL problem is a **policy** π
  - Let $\pi_t(\mathbf{x}_t)$ be a **function** that maps a state $\mathbf{x}_t \in \mathcal{X}$ to a feasible control input $\mathbf{u}_t \in \mathcal{U}(\mathbf{x}_t)$
  - The sequence $\pi_{0:T-1} := \{\pi_0(\cdot), \pi_1(\cdot), \ldots, \pi_{T-1}(\cdot)\}$ of functions is called an **admissible control policy**
  - To simplify notation, we informally denote $\pi_{0:T-1}$ by π
  - The long-term cost (reward) $V_t^\pi(\mathbf{x}_t)$ of a policy π starting at time t at state $\mathbf{x}_t$ is called the **value function** of π:

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

  - A policy π^* is an **optimal policy** if $V_0^{\pi^*}(\mathbf{x}_0) \leq V_0^\pi(\mathbf{x}_0)$ for all admissible π and its value function is denoted $V_0^*(\mathbf{x}_0) := V_0^{\pi^*}(\mathbf{x}_0)$

Conventions and Observations

- The problem of acting optimally is called:
  - **Optimal Control** (OC): when the models p_f, p_h are known
  - **Reinforcement Learning** (RL): when the models p_f, p_h are unknown but samples can be obtained from them
  - **Inverse RL/OC**: when the cost (reward) functions ℓ, $\mathfrak{q}$ are unknown
- Conventions differ in optimal control and reinforcement learning:
  - **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

Conventions and Observations

- Goal: select controls to minimize long-term cumulative costs
  - Controls may have long-term consequences, e.g., delayed reward
  - It may be better to sacrifice immediate reward 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 opponent moves might help winning chances many moves from now

- A policy fully defines the behavior of a robot by specifying, at any given point in time, which controls to apply.

- Policies can be:
  - **stationary** ($\pi \equiv \pi_0 \equiv \pi_1 \equiv \cdots$) $\subset$ **non-stationary** (time-dependent)
  - **deterministic** ($\mathbf{u}_t = \pi_t(\mathbf{x}_t)$) $\subset$ **stochastic** ($\mathbf{u}_t \sim \pi_t(\cdot \mid \mathbf{x}_t)$)
  - **open-loop** (a sequence $\mathbf{u}_{0:T-1}$ regardless of $\mathbf{x}_t$) $\subset$ **closed-loop** (π_t depends on $\mathbf{x}_t$)

Problem Variations

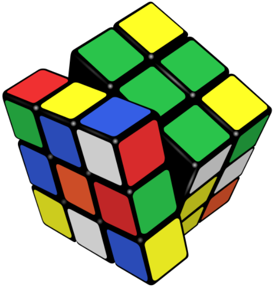
- **deterministic** (no noise $\mathbf{v}_t$, $\mathbf{w}_t$) vs **stochastic**
- **fully observable** ($\mathbf{z}_t = \mathbf{x}_t$) vs **partially observable** ($\mathbf{z}_t \sim p_h(\cdot|\mathbf{x}_t)$)
  - **fully observable**: Markov Decision Process (MDP)
  - **partially observable**: Partially Observable Markov Decision Process (POMDP)
- **stationary** vs **nonstationary** (time-dependent $p_{f,t}$, $p_{h,t}$, ℓ_t)
- **finite** vs **continuous** state space $\mathcal{X}$
  - tabular approach vs function approximation (linear, SVM, neural nets,...)
- **finite** vs **continuous** control space $\mathcal{U}$:
  - tabular approach vs optimization problem to select next-best control
- **discrete** vs **continuous** time:
  - finite-horizon discrete time: dynamic programming
  - infinite-horizon ($T \to \infty$) discrete time: Bellman equation (first-exit vs discounted vs average-reward)
  - continuous time: Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE)
- reinforcement learning (p_f, p_h are unknown) variants:
  - **Model-based RL**: explicitly approximate models $\hat{p}_f$, $\hat{p}_h$ from experience and use optimal control algorithms
  - **Model-free RL**: directly approximate V_t^* and π_t^* without approximating the motion/observation models

Example: Inventory Control

- Consider the problem of keeping an item stocked in a warehouse:
  - If there is too little, we will run out of it soon (not preferred).
  - If there is too much, the storage cost will be high (not preferred).
- We can model this scenario as a discrete-time system:
  - $x_t \in \mathbb{R}$: 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**: $x_{t+1} = x_t + u_t - w_t$
  - **Cost function**: $\mathbb{E}\left[R(x_T) + \sum_{t=0}^{T-1}\left(r(x_t) + cu_t - pw_t\right)\right]$ where
    - pw_t: revenue
    - cu_t: cost of items
    - $r(x_t)$: penalizes too much stock or negative stock
    - $R(x_T)$: remaining items we cannot sell or demand that we cannot meet

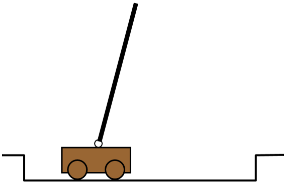
Example: Rubik's Cube

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



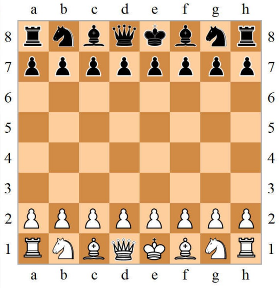
Example: Cart-Pole Problem

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



Example: Chess

- Formalization:
  - State space: $\sim 10^{47}$
  - Actions: from 0 to 218
  - Reward: 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 is 10^{123}



Example: Grid World Navigation

- Navigate to a goal without crashing into obstacles
- Formalization:
  - State space: robot pose, e.g., 2-D position
  - Actions: allowable robot movement, e.g., $\{left, right, up, down\}$
  - Reward: -1 until the goal is reached; $-\infty$ if an obstacles is hit
  - Can be deterministic or stochastic; fully or partially observable