236609 - AI and Robotics - Fall 2024

Lesson 2a: Three-Layered Decision Making in Robotics

Sarah Keren

The Taub Faculty of Computer Science
Technion - Israel Institute of Technology

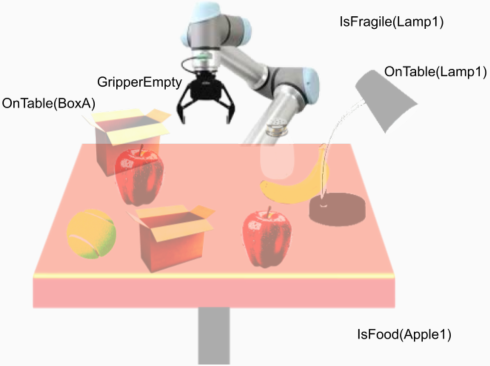
Table of contents

A Running Example

Example

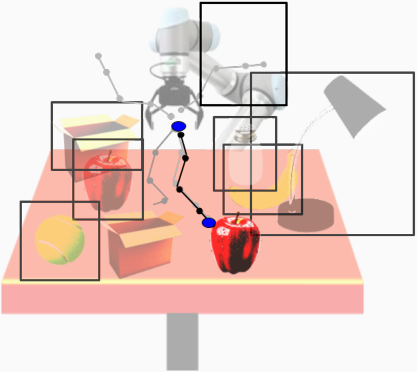
The person: ”I an hungry”

Task Planning

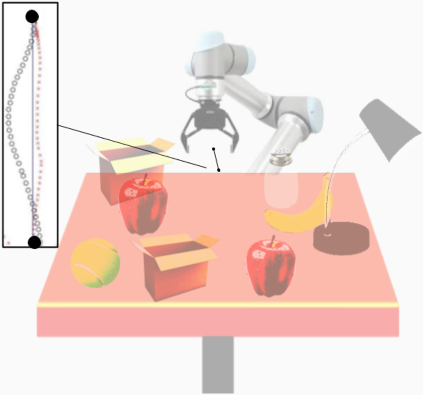


The person: ”I an hungry”

Motion Planning



Control

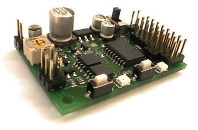


3 Layers of Decision Making

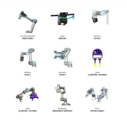
Components of a Robotic Agents

Sensors

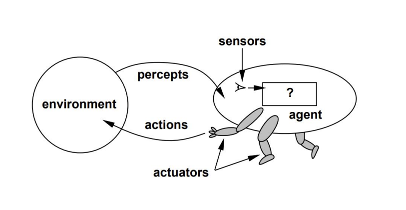
Controllers



Actuators

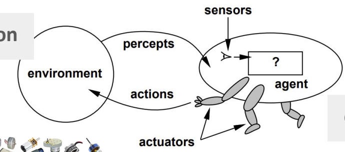


Autonomy



Autonomy

state estimation



decision making

motion control

Autonomy

State Estimation

$$\beta : \mathcal{S} \mapsto [0,1]$$

Process incoming observations to maintain a *belief* as a probability distribution over states

Decision Making

$$\pi : \beta \mapsto \mathcal{A}$$
$$\pi : \beta \times \mathcal{A} \mapsto [0,1]$$

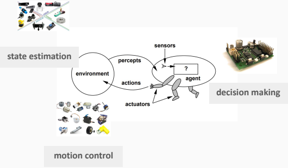
Find a policy - a mapping belief and objective into actions (probabilities)

Motion Control

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

Translate actions into low-level commands (and monitor their execution)

- $x(t) \in \mathbb{R}^n$ - state vector
- $u(t) \in \mathbb{R}^m$ - control input
- $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ - system dynamics
- $w(t)$ - noise or disturbance



Layered Control Architecture (LCA)

Semantic Logic Discrete Planning	**Decision Making**	Flexible and Slow
Optimization Sampling Methods Continuous Planning	**Trajectory Planning**	Intermediate
PID Control CLFs/CBFs	**Feedback Control**	Rigid and Real Time

Information Flow

Top-Down Flow:

- Goals → Plans → Commands
- Like company policies becoming specific actions

Bottom-Up Flow:

- Sensor data → Status updates → Results
- Like workers reporting to management

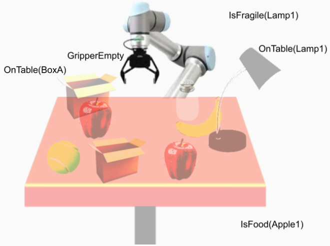
Time Scales

Different Speeds for Different Needs

- Planning: Seconds to minutes
- Behavioral: Fraction of seconds
- Execution: Milliseconds
- Control: Microseconds
- Hardware: Nanoseconds

Task Planning

What do we need to model ?

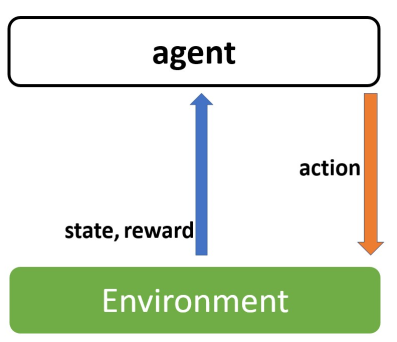


A robot controlled by a model-based AI program

- Model of the robot
- Model of the world
- Model of possible actions
- Model of other agents

AI program reasons about the model to make decisions

Modeling the Environment



What do we need to model ?

Modeling the Environment

What do we need to model ?

Modeling the Environment

What do we need to model ?

- Sequential decision making
- Uncertainty in action outcomes
- Partial observability

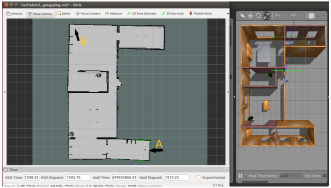
Additional considerations ?

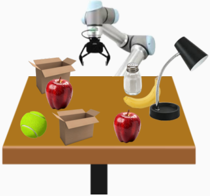
Model needs to be compact enough to be solvable but informative enough to be useful.

Environment: State and action space

- **State Space:** $s \in \mathcal{S}$ is a world state (more accurately, a representation of a world state)
  - Typically, it is impossible / impractical to explicitly maintain the state space.
  - We therefore use a **factored representation** in which the state space is described via a set of variables $\mathcal{X} = X_1, \ldots, X_n$, and a state is an assignment of a value $X_i \in Dom(X_i)$ for each variable X_i.
- **Action space:** $a \in \mathcal{A}$ is an action agents can perform
  - can have deterministic / non-deterministic / stochastic effects
  - may be associated with preconditions and effects.

Examples

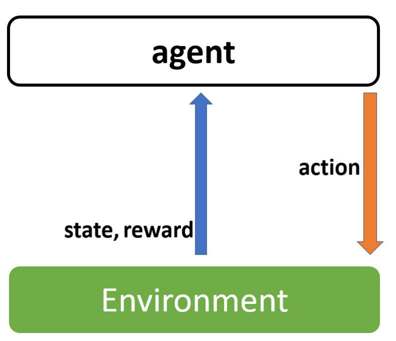




What are states and actions ?

The Agent-Environment Interface

- An agent operates in the environment by taking actions.
- An agent (and its behavior) is characterized by its:
  - **Objectives (reward and utility)**
  - Ability to **observe** and **sense** the environment (observability and beliefs)



The Agent-Environment Interface

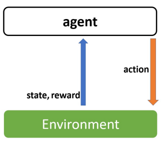
The agent and environment interact at each of a sequence of discrete time steps $t = 1, 2, 3, \ldots$.
At each time step t:

- The agent receives some representation of the environment's state - $s_t \in \mathcal{S}$
- On that basis selects an action $a_t \in \mathcal{A}(s)$

One time step later $t + 1$:

- As a consequence of its action, the agent receives a numerical reward $r_{t+1} \in R$
- Finds itself in a new state s_{t+1}

The environment and agent together thereby give rise to a sequence or **trajectory**: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots$

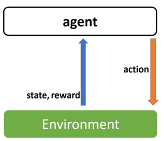


The Agent-Environment Interface

- **Episodes:** when the agent–environment interaction breaks naturally into sub-sequences. Each episode ends in a special state called the *terminal state*.
- **Episodic tasks:** Tasks with episodes. The time of termination T is a random variable that normally varies from episode to episode.
- **Continuing tasks:** When the agent–environment interaction does not break naturally into identifiable episodes, but goes on continually without limit.

Real-world examples of episodic vs. continuing tasks ?

Reward



- **Reward:** a signal passed from the environment to the agent (typically referred to as **cost** when reward is negative).
- A reward r_t is a scalar feedback signal Indicates how well agent is doing at step t.
- The reward signal is your way of communicating to the agent *what* you want achieved, not *how* you want it achieved.

https://deepmind.com/research/publications/2021/Reward-is-Enough

Reward (cont.)

- Often described by a reward function:
  - Depending on the setting, defined as $\mathcal{R}(s, a, s')$, $\mathcal{R}(s, a)$ or $\mathcal{R}(s)$
  - The reward can be stationary $\mathcal{R}(s, a, s') \in \mathbb{R}$ or non-stationary, in which case we consider the expectation $\mathcal{R}(s, a, s') = \mathbb{E}[r_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$
- An agent's objectives is to maximize it's **utility** $\mathcal{U}$
  - the expected total reward (*return*) it receives
  - the min reward
- Utility is sometimes known as *goal* - but we will refer to a goal as a state (or state set) an agent aims to reach.

https://deepmind.com/research/publications/2021/Reward-is-Enough

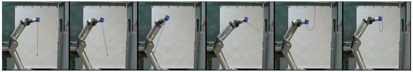
Reward (by David Sliver)

- Fly stunt manoeuvres in a helicopter
  - + for following desired trajectory
  - − for crashing
- Defeat the world champion at Backgammon
  - +/− reward for winning/losing a game
- Manage an investment portfolio
  - + reward for each $ in bank
- Control a power station
  - + reward for producing power
  - - reward for exceeding safety thresholds
- Make a humanoid robot walk
  - + reward for forward motion
  - - reward for falling over
- Play many different Atari games better than humans
  - +/− reward for increasing/decreasing score

Problems with this approach ?

Reward

Designing a good reward function is an art.



Ball in cup image from Kober et al (2009)

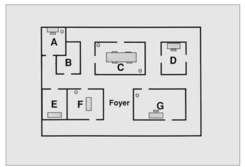
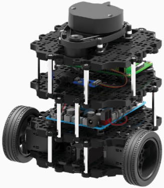
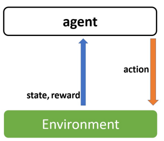
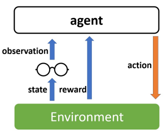


Figure 2. The Competition Arena.



Observation and Belief

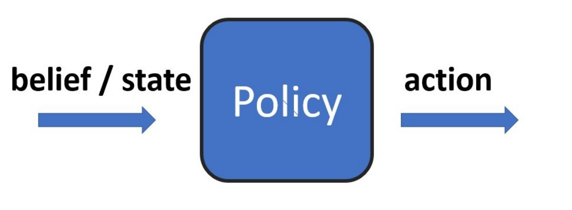




- At each time step the agent receives an observation o_t that reflects the current state of the world.
- A **belief** $\beta \in \mathcal{B}$ represents the possible world states (i.e., the states that are deemed possible by the agent).
- Induced by an **observability/ sensor function** that maps states to observations (more on this later)

Policy (for a single agent[1])

- A policy is a mapping from states to actions.
- More accurately: we seek a mapping from the agent's understanding of the state, i.e., its **belief**, to actions.
- Our objective is to find a **(sub)-optimal policy** for an agent to follow in order to achieve it's objective / maximize its **utility**.



[1]we will consider multi-agent policies in the second part of the course

Formal definitions: Policy

- A **deterministic** policy is a mapping:
  - $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from states to actions
  - $\pi : \mathcal{B} \rightarrow \mathcal{A}$ from beliefs to action
  - $\pi(s)$/ $\pi(\beta)$ is the (single) action the agent will perform at state s or belief β.
- A **non-deterministic** policy is a mapping:
  - $\pi : \mathcal{S} \rightarrow \mathcal{A}^n$ from states to actions,
  - $\pi : \mathcal{B} \rightarrow \mathcal{A}^n$ from beliefs to actions
  - $\pi(s)$/ $\pi(\beta)$ is the set of actions one of which the agent will perform at state s or belief β.
- A **stochastic** policy is a mapping:
  - $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ from states to actions
  - $\pi : \mathcal{B} \times \mathcal{A} \rightarrow [0, 1]$ from beliefs to action
  - $\pi(s, a)$/ $\pi(\beta, a)$ is the probability the agent will perform action a at state s/ belief β.

Policies for Reactive Agents

SIMPLE ROOMBA

If BUMP = TRUE
THEN Turn (random direction/amt)

ELSE MOVE-STRAIGHT

https://www.youtube.com/watch?v=7FSUtSurqA4

Policies for Rational Agents

- Typically cannot be described explicitly for each state and require a compact representation.
- An **optimal policy**, denoted π^* is *better* than or equal to all other policies.
- Since our focus is on rational agents with a clearly defined utility measure, an optimal policy maximizes the agent's utility function $\mathcal{U}$.

Learning from a sample set

In supervised learning:

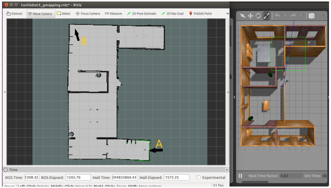
- A sample set: $S = \{(x_i, y_i)_{i=1}^{m}\}$
- Typically used to learn for an appropriate classifier (e.g., with lowest expected error or loss) among a hypothesis class H.

Learning a policy π:

- A sample set: $S = \{(s_i, a_i)_{i=1}^{m}\}$ or $S = \{(\beta_i, a_i)_{i=1}^{m}\}$
- Typically, a parameterized policy over a factored state-space representation is used.
- Parameterized policy as a conditional probability $\pi_\theta(a_t|\beta_t)$ or $\pi_\theta(a_t|s_t)$ (for fully observed)

Sample set representation of our domains ?

A sample set: $S = \{(s_i, a_i)_{i=1}^{m}\}$ or $S = \{(\beta_i, a_i)_{i=1}^{m}\}$



Effective for sequential decision making ?

Propositional STRIPS

Propositional STRIPS planning task defined by a tuple $P = \langle \mathcal{F}, I, \mathcal{A}, G, \mathcal{C} \rangle$, where

- $\mathcal{F}$ is a set of fluents and a state s is represented by the fluents that are true in s
- $I \subseteq F$ is the initial state
- $G \subseteq F$ represents the set of goal states, and
- $\mathcal{A}$ is a set of actions.
  - Each action is a triple $a = \langle pre(a), add(a), del(a) \rangle$, which represents the precondition, add, and delete lists respectively, all subsets of $\mathcal{F}$.
  - An action a is applicable in state s if $pre(a) \subseteq s$.
  - If action a is applied in state s, it results in a new state $s' = (s \setminus del(a)) \cup add(a)$.
- $\mathcal{C} : \mathcal{A} \to \mathbb{R}_0^+$ is a cost function that assigns each action a non-negative cost.

Propositional STRIPS

- The objective is to find a plan $\pi = \langle a_1, \ldots, a_n \rangle$: a sequence of actions that brings an agent from the initial state to a goal state.
- The cost $c(\pi)$ of a plan π is $\Sigma_{i=1}^{n}(C(a_i))$.
- Often, the objective is to find an optimal solution for P: a plan π^* that minimizes the associated cost.
- The literature is rich with different approaches developed to solve the planning problem (Bonet and Geffner 2013): more on this later on in the course.

From *https://ai.dmi.unibas.ch/misc/tutorial_aaai2015/planning02.pdf*

Markov Decision Process

A **Markov Decision Process**(MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathcal{S}$ is a finite set of states defined via a set of random variables $\mathcal{X} = X_1, \ldots, X_n$
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix $\mathcal{P}^a_{s,s'} = \mathcal{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$, and
- **optional:** γ is a discount factor $\gamma \in [0, 1]$ that is used to favor immediate rewards over future rewards.

The Markov property: "The future is independent of the past given the present".

Extensions: Infinite and continuous MDPs, partially observable MDPs, undiscounted, average reward MDPs. etc.

Rewards (from Sutton and Barto 2018)

- Expected return for episodic tasks:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T = \sum_{k=0}^{T} r_{t+k+1}$$

- Expected return for continuing tasks:

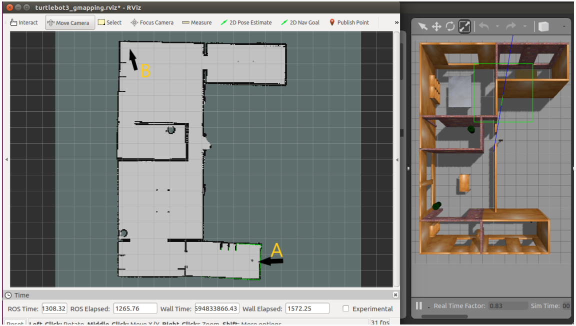
$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

  when γ is the discount factor.

- Returns at successive time steps are related to each other:

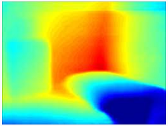
$$G_t = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots) = r_{t+1} + \gamma G_{t+1}$$

How to model our domain as MDPs ?

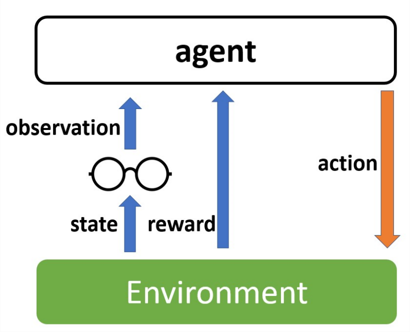


Is this model appropriate? Limitations? Strengths?

Accounting for Partial Observability



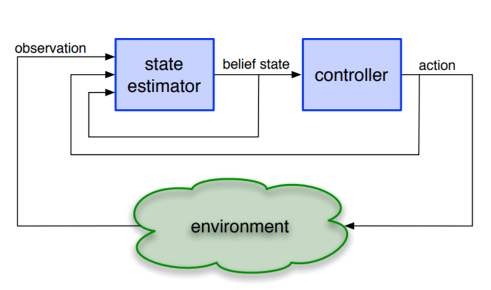
Reminder: Observation and Belief



At each time step the agent receives an observation o_t that reflects the current state of the world. s

Beliefs and Belief Tracking

- A **belief** is a representation of the possible world states.
- In partially observable domains, we may have a **sensor model** represented as a mapping function from what is observed to the actual world state.
- The agent maintains its belief via a *state estimator* - which we will refer to as the process of **Belief Tracking**.



From Kaelbling, L. P., and T. Lozano-Perez. "Integrated Task and Motion Planning in Belief Space" 2013 *https://dspace.mit.edu/bitstream/handle/1721.1/87038/Kaelbling_Integrated%20task.pdf?sequence=1&isAllowed=y*

Belief

A belief $\beta \in \mathcal{B}$ can have different representations:

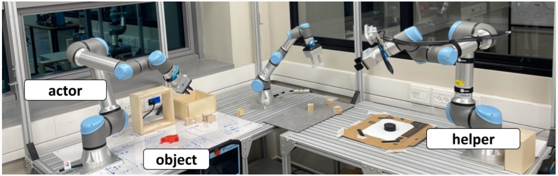
- **Deterministic:** a direct mapping from states to beliefs $\beta = s$ (in which case we simply talk about states)
- **Non-deterministic:** $\beta \subseteq \mathcal{S}$ represents the set of possible world states
- **Stochastic:** the belief is a probability distribution over possible underlying world states. $\beta : \mathcal{S} \to \mathcal{P}[\mathcal{S}]$ such that $\beta(s)$ represents the probability that s is the actual world state.

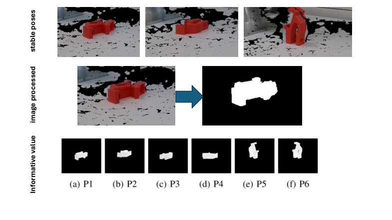
Recall that the environment-agent interactions give rise to a sequence or trajectory: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots$

From the point of view of the partially informed agent, we have a **history**: $o_0, a_0, r_1, o_1, a_1, r_2, o_2, a_2, r_3, \ldots$

The agent needs to maintain its belief based on the current observation.

Beliefs in the Lab





$$\beta^{o,s}(p) = \frac{\hat{P}(o|p,s)\,\beta(p)}{\int_{p' \in \mathcal{P}_o} \hat{P}(o|p',s)\,\beta(p')\,dp'} \tag{1}$$

where $\beta(p)$ is the estimated probability that p is the pose prior to considering observation o.

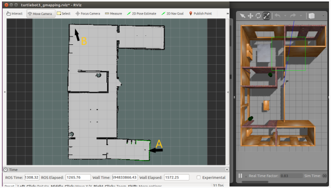
Partially Observable Markov Decision Process (POMDP)

A **Partially Observable Markov Decision Process**(POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O}, \beta_0 \rangle$ where

- $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$ and γ are as for an MDP.
- Ω is a set of observations (observation tokens),
- $\mathcal{O}$ is a sensor function specifying the conditional observation probabilities $\mathcal{O}^{o}_{s,a} = \mathcal{P}[O_{t+1} = o | S_t = s, A_t = a]$ of receiving observation token $o \in \Omega$ in state s after applying action a [2].
- β_0 the initial belief: a probability distribution over the states such that $\beta_0(s)$ stands for the probability of s being the true initial state.

[2]alternatively: $\mathcal{O}^{o}_{s} = \mathcal{P}[o_t = o | S_t = s]$

How to model our domain as a POMDP ?



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O}, b_0 \rangle$

Is this model appropriate? Limitations? Strengths?

POMDPs at the CLAIR lab

$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O}, b_0 \rangle$

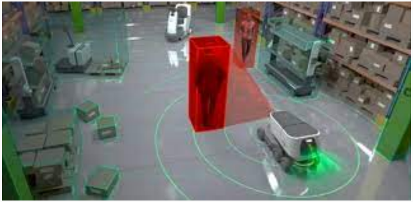
- State space: Position and orientation of all objects and robot configuration $\mathcal{S} = \{(x_i, y_i, z_i, rx_i, ry_i, rz_i)\}_{i=1}^{N_{items}} \bigcup \mathcal{C}_{robot}$
- Action space: pick up or sense if there's object for each point in the workspace $\mathcal{A} = \{sense(x, y) | x, y \in workspace\} \bigcup \{sense(x, y) | x, y \in workspace\}$
- Observation Space: whether an item is picked at the last action or whether an object was sensed at the last action $\{picked, not_picked, sensed, not_sensed\}$

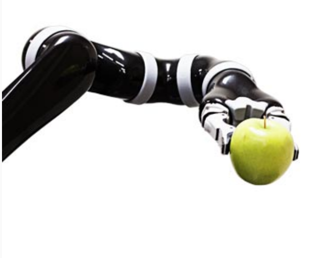
Optimization Considerations

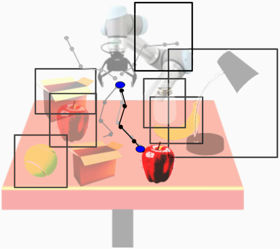
- Completeness
- Soundness
- Optimality
- Anytime guarantees
- Computational Efficiency
- Generality
- Memory consumption

Motion Planning

Motion Planning







Motion Planning

From **Oren Salzman's** course slides*

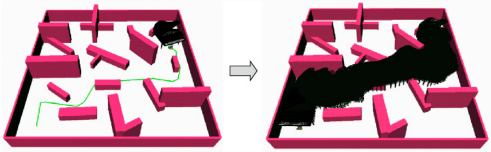
(Formal) definition of the basic motion-planning problem**
Let $\mathcal{R}$ be a robot system with d **degrees of freedom**, moving in a known environment cluttered with obstacles. Given start and target **configurations** s and s' for $\mathcal{R}$, decide whether there is a collision-free, continuous path $\tau : [0, 1] \to \mathcal{C}_{free}$ such that $\tau(0) = s$ and $\tau(1) = s'$ and if so, plan such a motion.

---

*Consider taking course 236901 to learn about algorithms for robotic motion planning.

https://arxiv.org/pdf/2209.14471.pdf

Motion Planning is Hard



More about this soon...

Control

Example of a Control Problem: Follow the Motion Plan

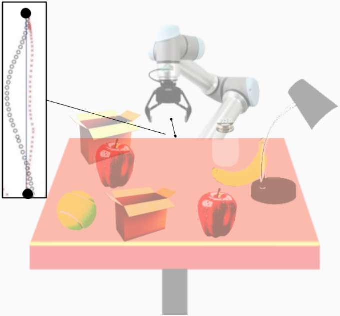
Given a motion plan τ, the trajectory tracking problem is defined by (X, U, f, τ) where:

- X is the state space (can be the configuration space and the derivatives in that space (velocities))
- U is the control/input space
- $f : X \times U \to X$ is the system dynamics what is the next state given the current and the control input, what assumption do we make here?
- $\tau : [0, 1] \to C_{free}$ is the reference motion plan

The control objective is to find $u(t)$ such that:

- $\dot{x}(t) = f(x(t), u(t))$ This is the system dynamic equation. Note that it's a ODE
- $\|x(t) - \tau(s(t))\| < \varepsilon$ for some tracking error ε We are close enough to the trajectory
- $s(t)$ maps time to path parameter $[0, 1]$

Control



More about this very soon...

Summary

Summary:

- We tool a closer look at the structure of a robot
- We examined different models used in AI and robotics

What next ?

- Cover basic reactive approaches to robotic control