236609 - AI and Robotics - Fall 2022

Lesson 3: A slightly deeper dive into the three decision-making layers

Sarah Keren

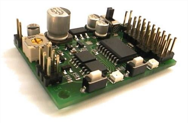
The Taub Faculty of Computer Science
Technion - Israel Institute of Technology

Table of contents

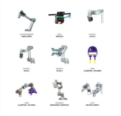
Components of a Robotic Agents

Sensors

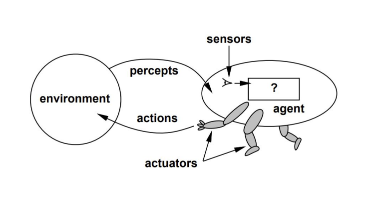
Controllers



Actuators

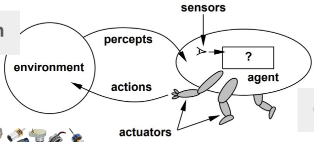


Autonomy



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

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

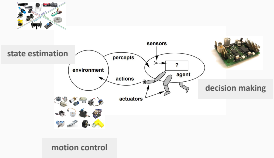
Find a policy - a mapping belief and objective into actions (probabilities)

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
- $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^n$ - 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

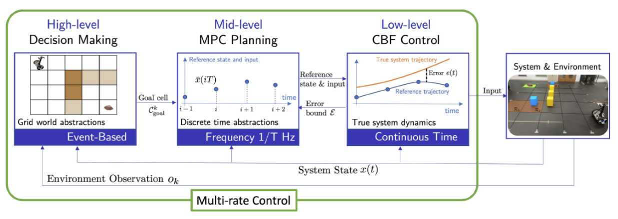
- Planning: Seconds to minutes
- Behavioral: Fraction of seconds
- Execution: Milliseconds
- Control: Microseconds
- Hardware: Nanoseconds

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

Matni, Nikolai, Aaron D. Ames, and John C. Doyle. "A Quantitative Framework for Layered Multirate Control: Toward a Theory of Control Architecture." IEEE Control Systems Magazine (2024)

Layered Control Architecture (LCA)



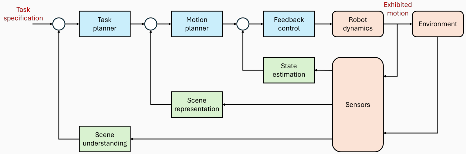
Matni, Nikolai, Aaron D. Ames, and John C. Doyle. ”A Quantitative Framework for Layered Multirate Control: Toward a Theory of Control Architecture.” IEEE Control Systems Magazine (2024)

Challenge: Robots need to:

- Make high-level decisions
- React to their environment
- Control precise movements
- Handle multiple tasks
- Respond at different speeds

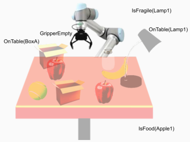
The Three Layers

Layered Control Architecture (LCA)



Task Planning

Top Layer: High level planning



Example:

- Realize that the human needs a food item.
- Reason about the reachability of items and the ability to perceive them.
- Decide which food object to pick-up.
- Optional - Prepare for failure via a contingent plan
- ...

What is a good model?

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
- $\mathcal{A}$ 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.

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 [1].
- 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.

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

Partially Observable Markov Decision Process (POMDP)

In its most general formulation, a belief β is represented as a probability distribution over the states $\mathcal{S}$, such that $\beta(s)$ stands for the probability of s being the true state.

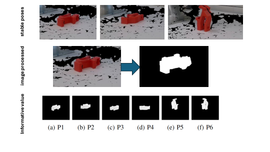


Figure 1: How to represent beliefs of robotic agents?

Typically, each agent is associated with a **belief update function** $\tau : \mathcal{B} \times \Omega \mapsto \mathcal{B}$ that represents how a belief changes when receiving a new observation.

POMDP for our example?

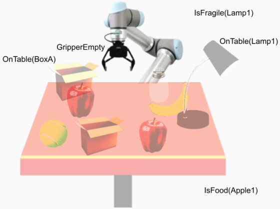


Figure 2: What is a good representation ?

YAY! POMDPs

- The POMDP provides a principled and formal framework for long-term decision making in settings in which it is important to account for the probabilistic nature of the sensor function.
- Has been successfully used in many robotic applications, including localization and navigation, search and tracking, autonomous driving, multi-robot systems, manipulation, and human-robot interaction.
- A key benefit: many planning tools that have been developed (most based on sampling and approximations).

POMDPs are hard! (and not always needed)

- Need to address the continuous nature of the environment through approximation and discretization techniques or through specialized algorithms designed for continuous domains.
- Computationally intractable - especially true for the high-dimentional robotic settings.

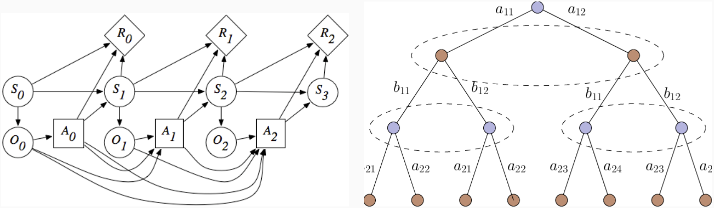


Figure 3: POMDP planning

Ideas?

Solution Approaches

- Use a simpler model when possible (e.g., MDP)
- Function approximations (AKA as Deep Learning)
- Sampling based approaches.
- Adopt a modular approach that decomposes the system into modules for state estimation, planning, and control

More on this later in the course !

Motion Planning

- Task planning focuses on achieving high-level goals and on finding sequences of high-level actions to achieve them
- Motion planning ensures the physical feasibility of these actions by generating **collision-free paths** or trajectories that respect the robot's **kinematic and dynamic constraints**.

Happens in a high-dimensional space!

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** q_0 and q_g for $\mathcal{R}$, decide whether there is a collision-free, continuous path $\tau : [0,1] \rightarrow \mathcal{C}_{free}$ such that $\tau(0) = q_0$ and $\tau(1) = q_g$ and if so, plan such a motion.

Alternatively, we can consider a *motion path* from initial configuration $q_0 \in \mathcal{C}$ to goal configuration $q_g \in \mathcal{C}$ as a sequence of configurations $(q_0, q_1, ..., q_m = q_0)$ where for all i, $\{\alpha q_{i-1} + (1-\alpha) q_i | \alpha \in [0,1]\} \subseteq \mathcal{C}_{free}$.

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

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

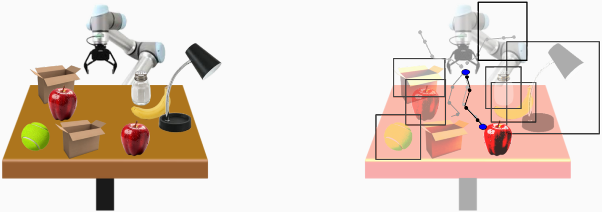


Figure 4: What is a good representation ?

Happens in a high-dimensional space!

Motion Planning

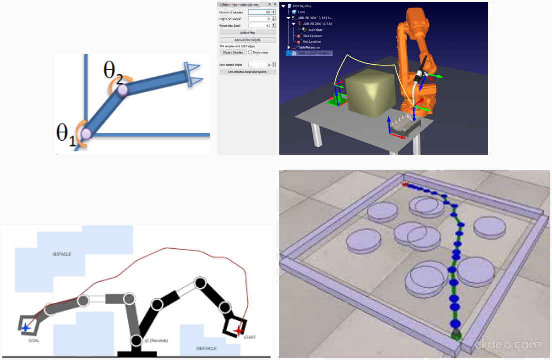


Figure 5: Motion Planning Examples

Happens in a high-dimensional space!

Motion Planning

- Typically, in order to support computationally feasibility, the path that the motion planner generates is a discretized abstraction of a path
- A path $\tau = \{w_i\}_{i=0}^{N}$ is is a sequence of waypoints w_i, where N denotes the total number of waypoints in τ, and w_0 and w_N represent the initial and goal configurations q_0 and q_g, respectively.

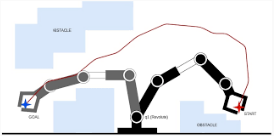
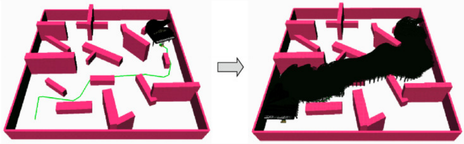


Figure 6: Motion Planning

Motion Planning is Hard



Motion planning approaches are diverse: optimization-based methods (e.g., trajectory optimization, gradient descent) sampling-based methods (e.g., Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM)), physics-based planners and hybrid approaches.

More on this later on ...

Feedback Control - Motion Control

Feedback Control

- Once a motion plan is found, there is a need to execute it.
- While motion planning aims to find the best path for the robot to follow,
- focus on ensuring that the robot follows this trajectory in the presence of real-world factors like sensor noise, dynamic disturbances, or actuator inaccuracies.

Feedback Control for our example?

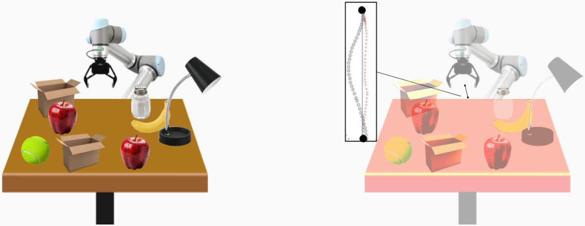
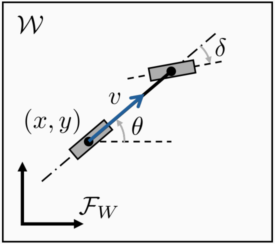
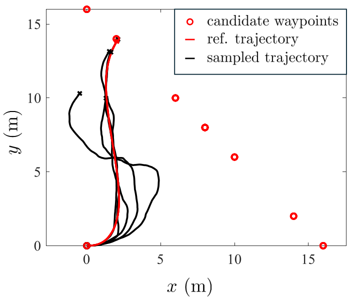


Figure 7: What is a good representation ?

Happens in a high-dimensional space!

Feedback Control





- To describe the control episode between consecutive waypoints w_{j-1} and w_j denoted by γ_j, we use a *dynamic model*.
- Let $\mathcal{F}_{\mathcal{W}}$ denote the inertial frame of reference embedded in $\mathcal{W}$, with origin $\mathcal{O}_{\mathcal{W}}$. The state $\mathbf{x}(k) \in \mathcal{R} \subset \mathbb{R}^d$ of the mobile agent is governed by the following difference equation,

$$\mathbf{x}(k+1) = f[\mathbf{x}(k) + \mathbf{B}[\mathbf{x}(k)]\mathbf{u}(k) + \mathbf{p}(k) \quad (1)$$

where $k = 0, \cdots, T$ is the index of time steps in the control episode, $\mathbf{u}(k) \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $\mathbf{B}[\mathbf{x}(k)]$ is the control effect matrix, and the vector $\mathbf{p}(k)$ represents process noise.

- We refer to the deterministic system with $\mathbf{p}(k) \equiv 0$ as the nominal model.

Feedback Control

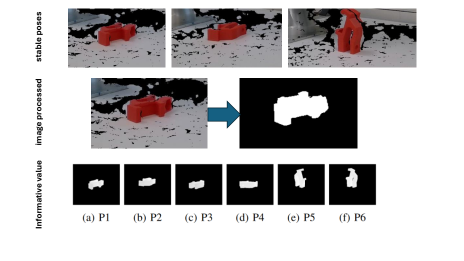
- Given the dynamic model, the challenge is to design a control system to set the behavior of a dynamic system in a desired way.
- This involves determining a sequence of control inputs $\mathbf{u}(k)$ to achieve specified states while satisfying system constraints, and dynamically adjusting them based on the current state.
- Many model-based approaches (e.g., PID, LQR, MPC) and many emerging techniques based on neural network-based control.

Putting it All Together

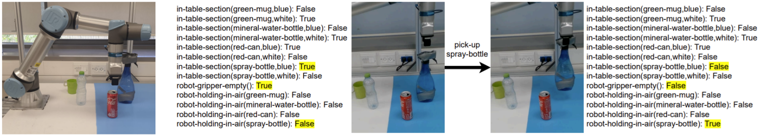
3-tiered State Estimation

- The account so far considers external or internal factors that affect the system dynamics but are not directly controlled or predicted by the model.
- In some settings, it is beneficial to explicitly model sensor noise, and the mapping between states and observations.
- State estimation needs to be performed for the 3 layers of abstraction:
  - What is the value of POMDP features ?
  - What is the position of the robot w.r.t its motion plan?
  - What is the position of the robot w.r.t its nominal path to the next waypoint?

State Estimation at the CLAIR Lab



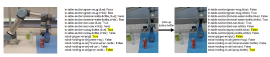
State Estimation at the CLAIR Lab



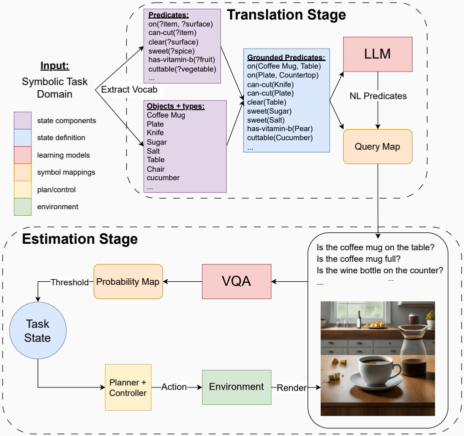
(a) Setup.

(b) Example transition annotated with S3E. State changes highlighted.

State Estimation at the CLAIR Lab



(a) Setup. (b) Example transition annotated with S3E. State changes highlighted.



3-tiered State Estimation

For task planning, we have a probabilistic sensor function where Ω is a set of observations and $\mathcal{O}$ 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 .

At the lower levels, a *sensor function* is used to record the position of the mobile agent in the inertial frame $\mathcal{F}_{\mathcal{W}}$ as

$$\mathbf{z}(k) = h[\psi(k)] + \mathbf{q}(k). \tag{2}$$

where, h is the expected observation of the system state $\psi(k)$ and $\mathbf{q}(k) \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the zero-mean i.i.d. Gaussian sensor noise.

Putting it all Together

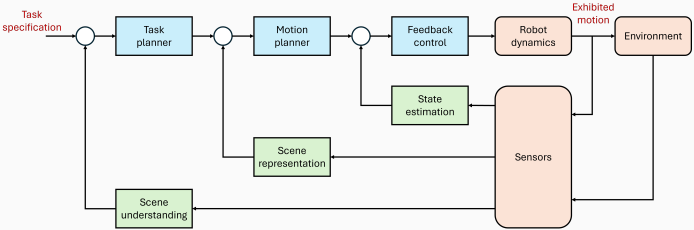


Figure 9: Full Process

Putting it all Together

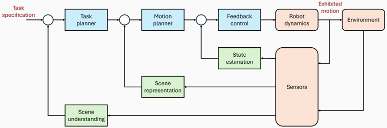


Figure 10: Caption

Different Time Scales for Different Needs

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

Information Flow:

- **Top-Down Flow:** Goals → Plans → Commands
- **Bottom-Up Flow:** Sensor data → Status updates → Results

Any ideas for information flow?

Summary

Summary:

- We looked at the different layers of decision-making

What next ?

- Take a closer look at feedback control