AAAI 2025 Tutorial T04
Time: 2025-02-25 8:30-12:30
Location: Room 118A

Part II: Foundation Models meet Physical Agents

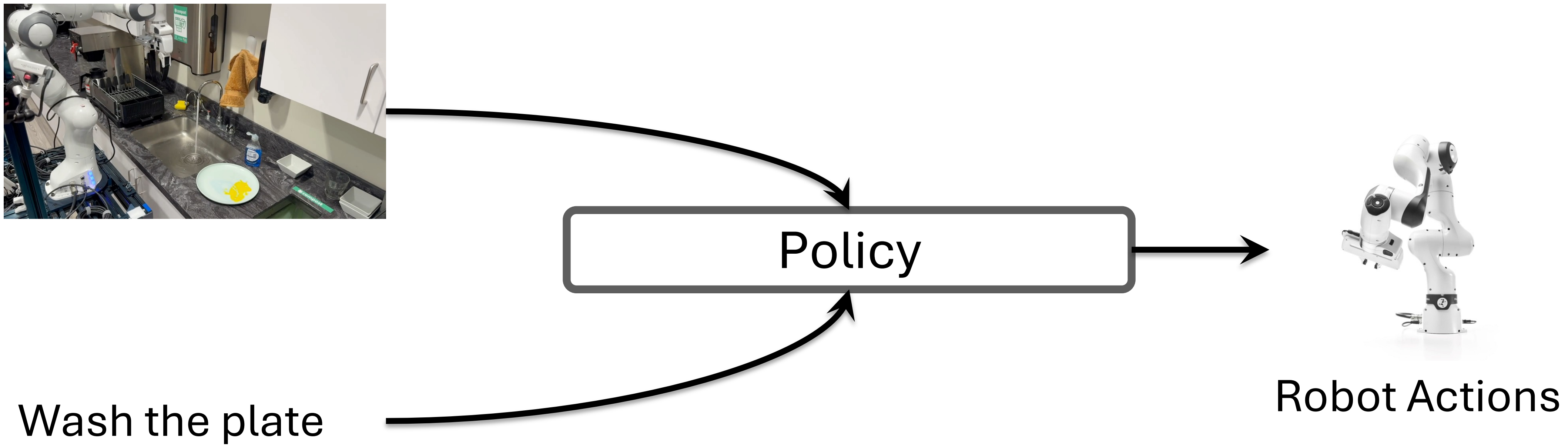
AAAI Tutorial: Foundation Models Meet Embodied Agents

Northwestern University
Columbia
MIT
Stanford University

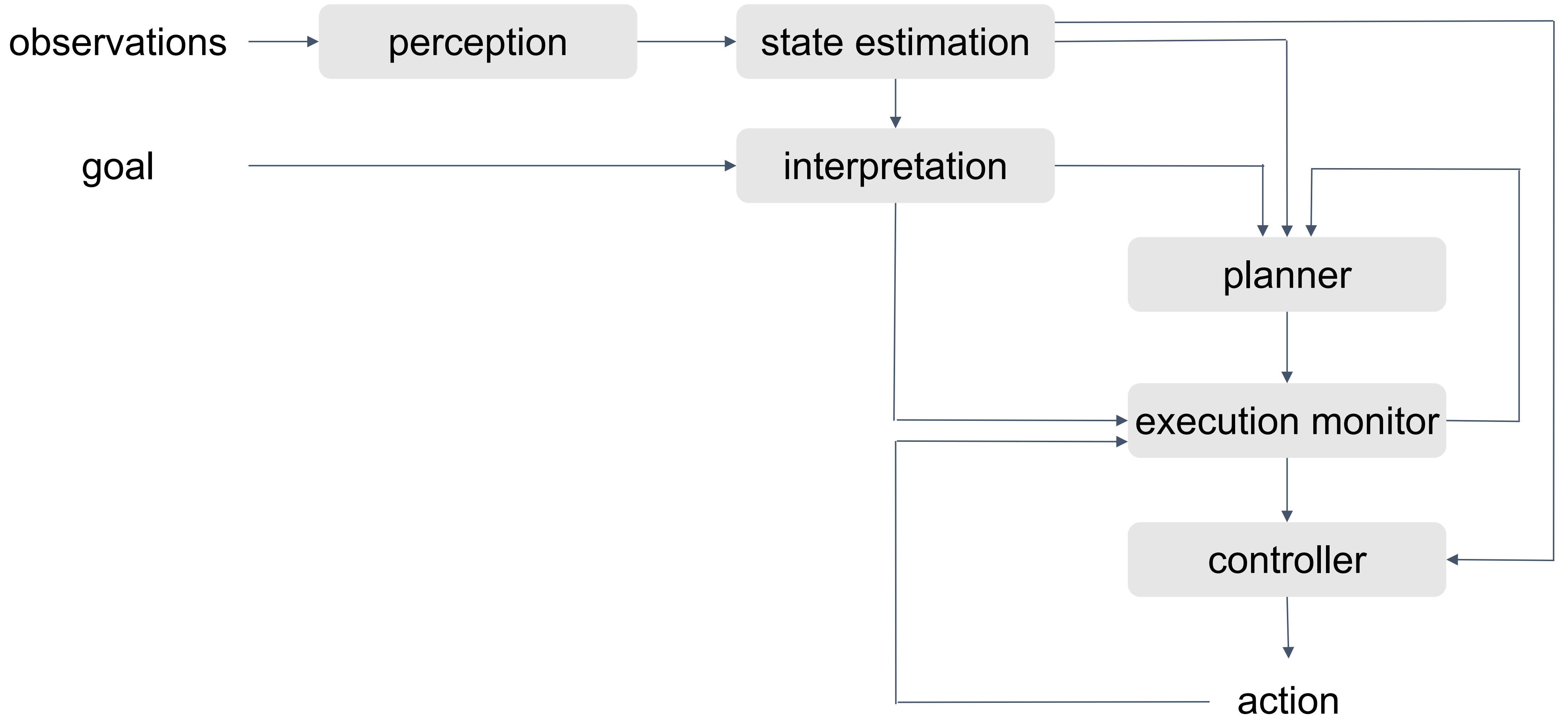
Physical Agents Overview

Physical Agents Overview

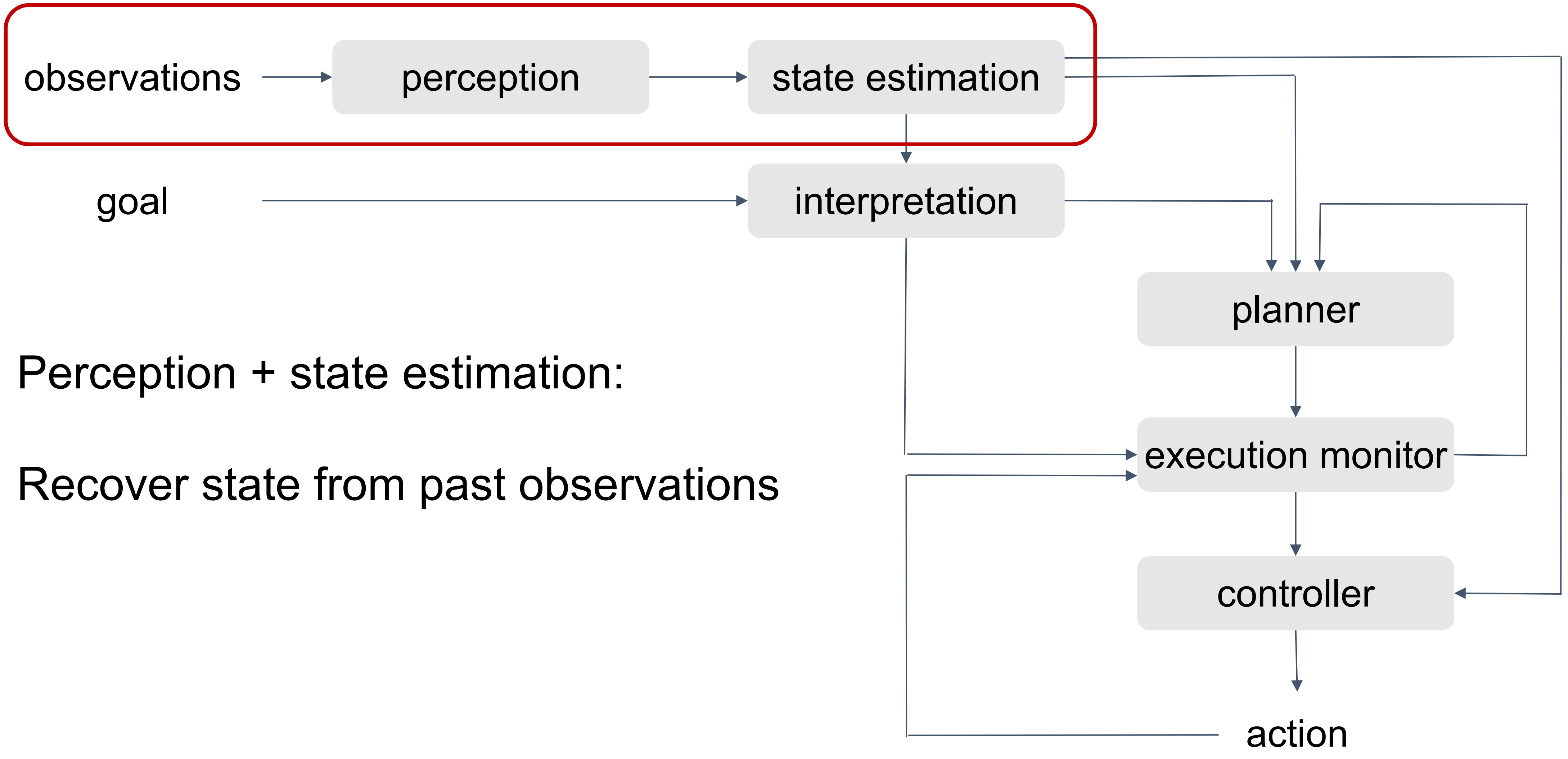
- Policy: $\pi(o, g) \rightarrow a$
- o: observation (images, robot proprioception, tactile, ...)
- g: goal (natural language for this tutorial)
- a: robot control commands



The Robot Architecture



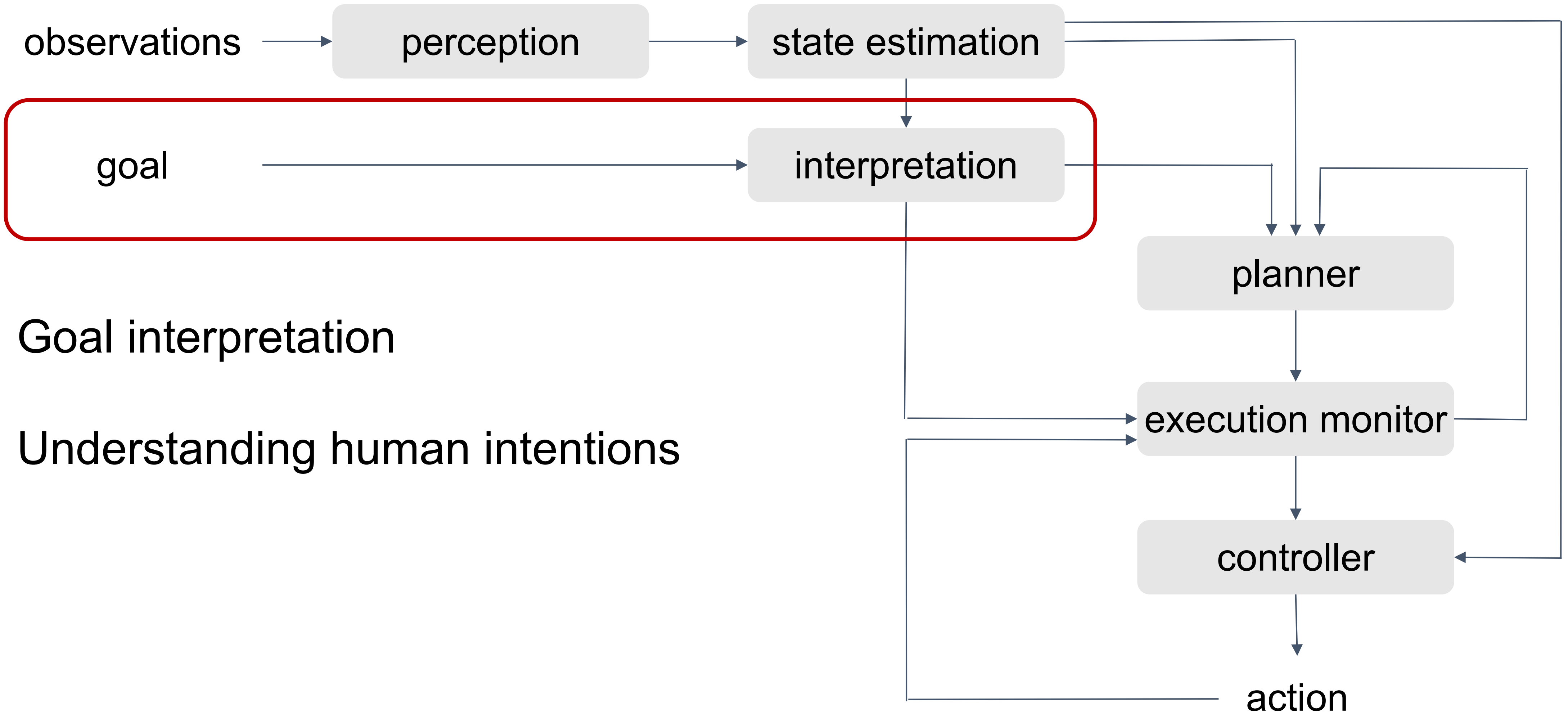
The Robot Architecture



Perception + state estimation:

Recover state from past observations

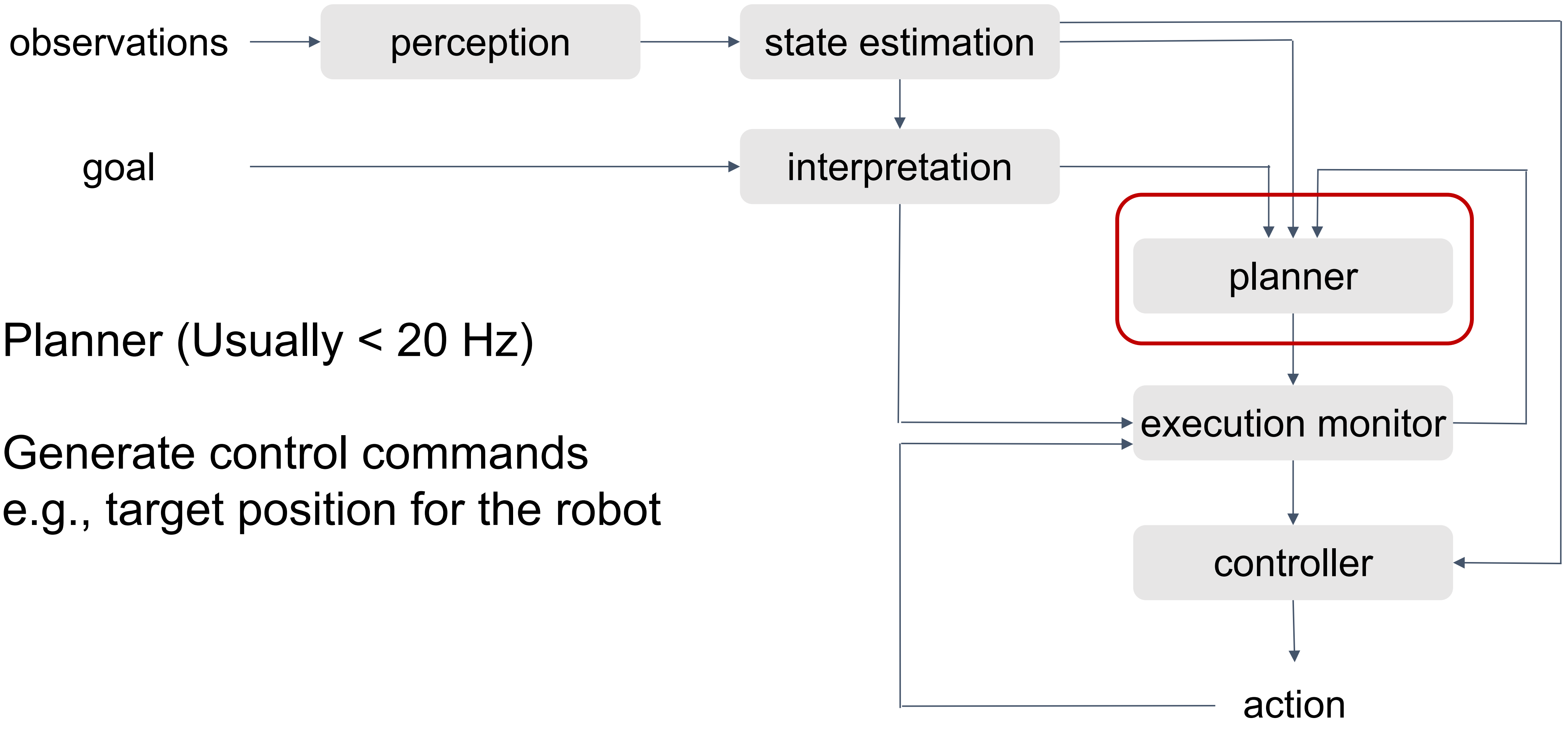
The Robot Architecture



Goal interpretation

Understanding human intentions

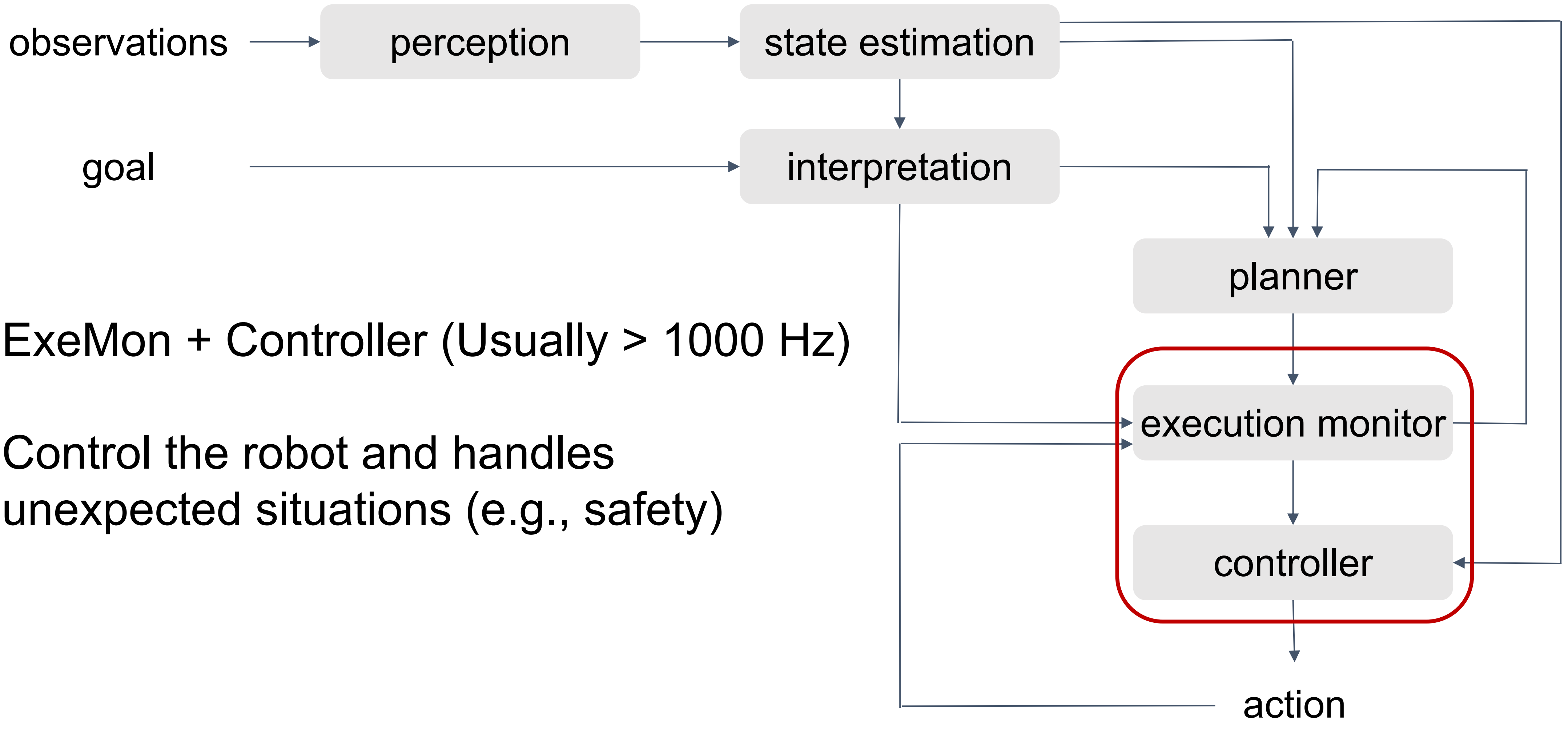
The Robot Architecture



Planner (Usually < 20 Hz)

Generate control commands
e.g., target position for the robot

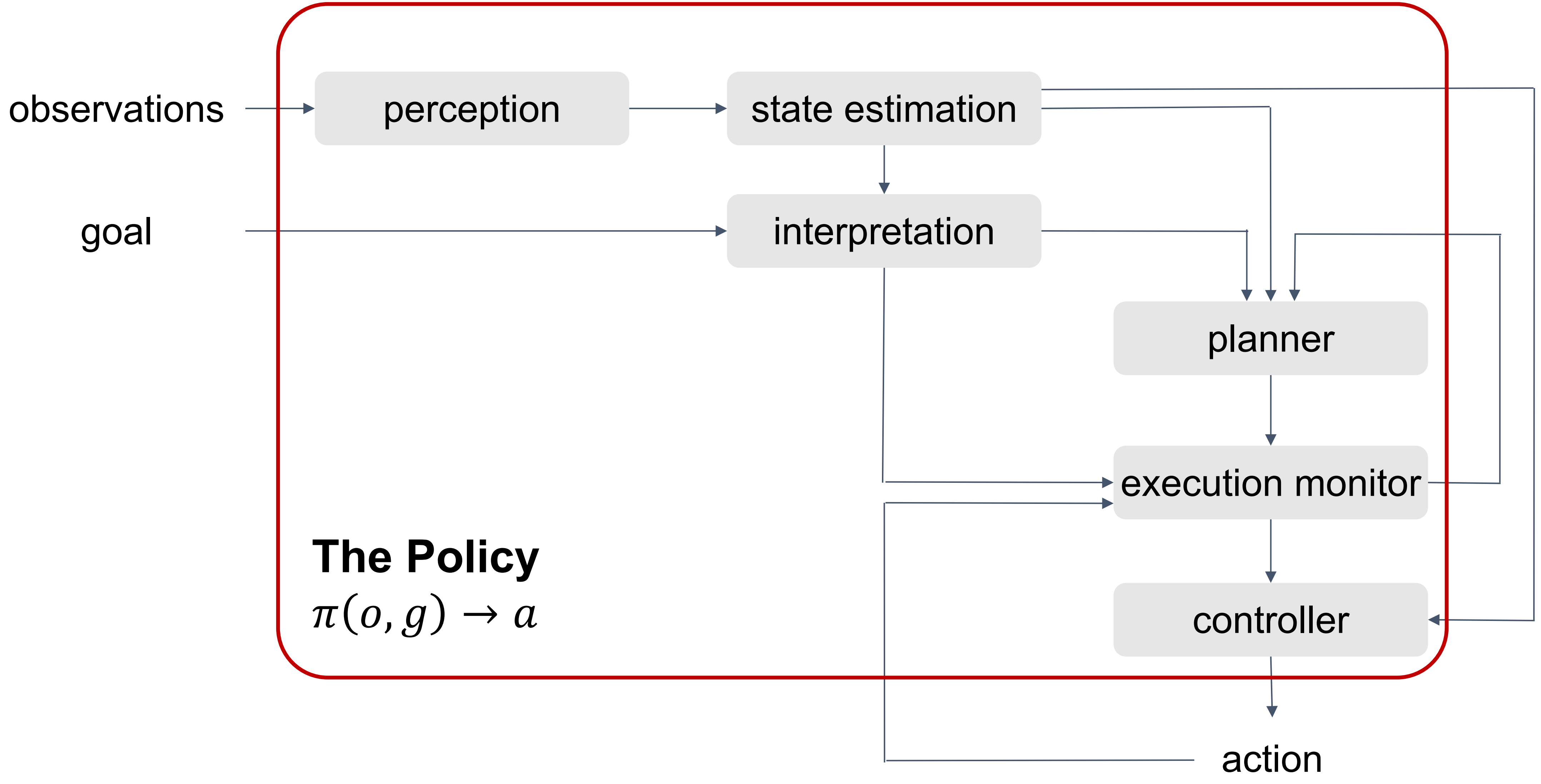
The Robot Architecture



ExeMon + Controller (Usually > 1000 Hz)

Control the robot and handles unexpected situations (e.g., safety)

The Robot Architecture



Key Questions Physical Agents

- How to design and build the “state”
  - Usually involves computer vision and signal processing techniques
  - Partial observability is very salient
- How to design and build the “action”
  - Usually involves both discrete and continuous parameters
- How to design and build policies (high-level and low-level)
  - High-level: primitive functions such as pick and place
  - Low-level: primitive control commands such as target position and velocity
- How to design and build transition models and reward functions
  - Ground-truth is unknown
  - Reward functions are usually hard to define manually
  - Reward functions also need to consider human preferences

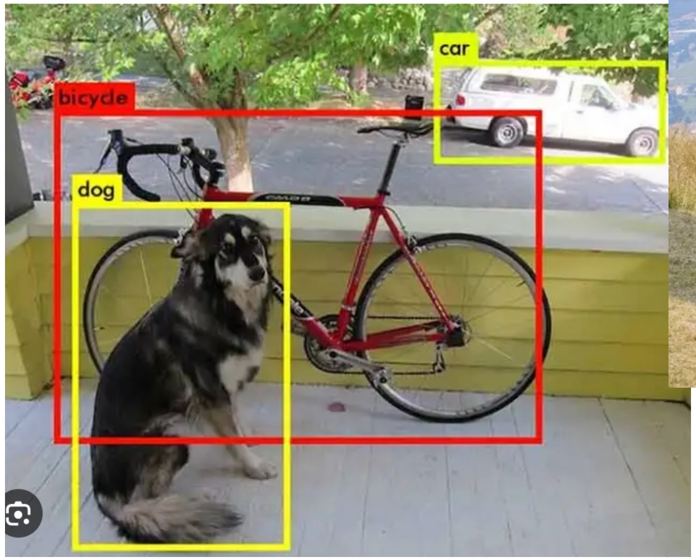
From Observations to States

- How to design and build the "state"
  - Usually involves computer vision and signal processing techniques
  - Partial observability is very salient

- In this tutorial, we will focus on obtaining representations about "objects"

Vision Techniques

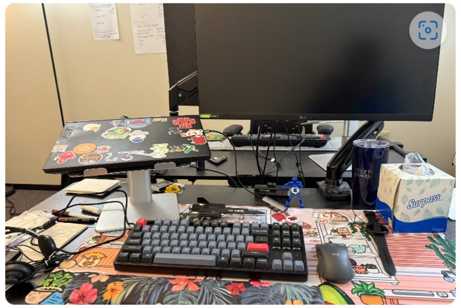
Segmentation



Detection

Tracking

Image-to-3D



describe the objects in this image -- just give a one word generic prompt for each object that you see in this image.

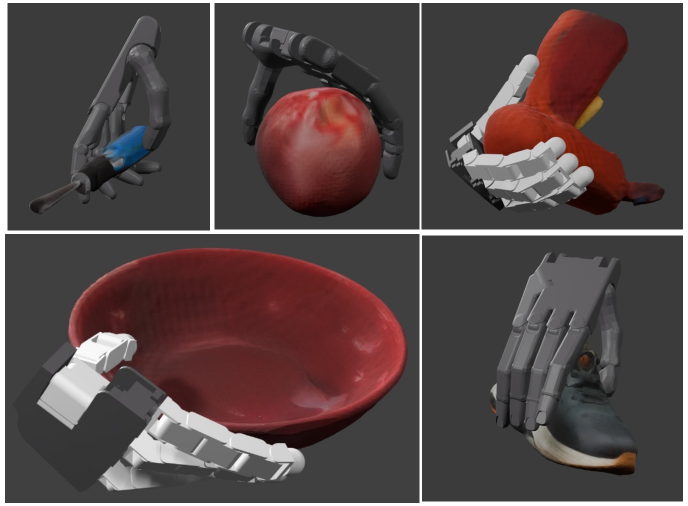
laptop, monitor, keyboard, mouse, cup, tissues, watch, cables, headphones, notebook, pen, stand, lamp, papers, toy, desk, chair

VLMs

3D Scene Understanding from Images

- **Accurate 3D scene understanding** is crucial for reliable manipulation in real environments

Grasping



Collision Avoidance

Reliable Placing

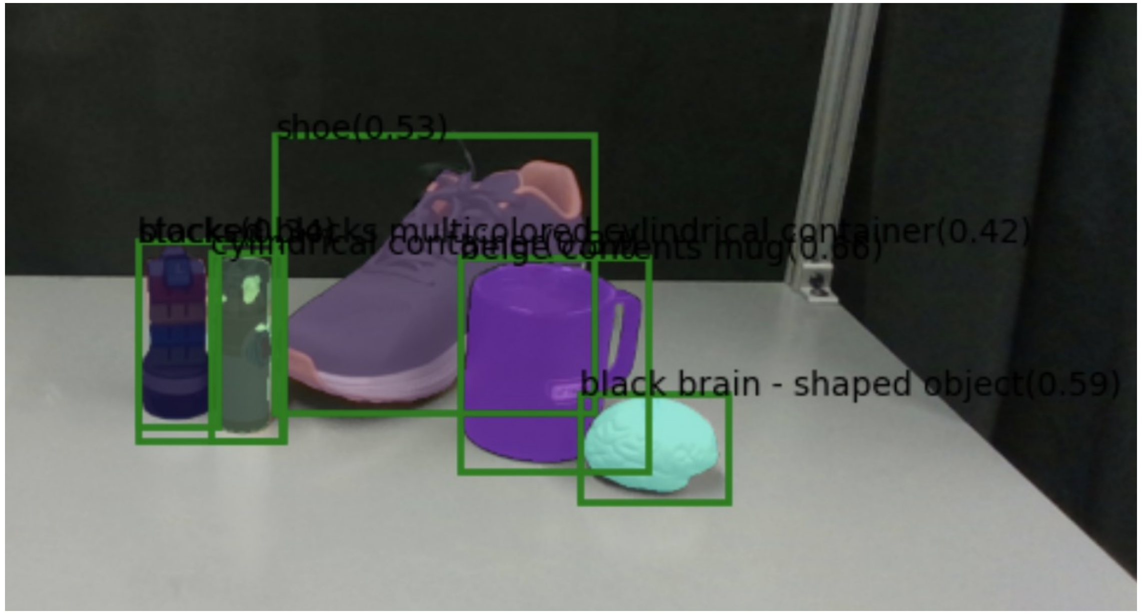
From Vision Models to Vision Foundation Models

- Different tasks were usually studied individually
- Different tasks rely on different datasets (e.g., fixed vocabulary object detection)
- Trend: Training on very large datasets for broad coverage

- Although they are called “vision FM” but they are designed to solve one particular task

Input: RGBD Captures

Object Detection



- Three commonly used object detection modules:
- Category-agnostic: Segment-Anything
- Category-specific: Mask-RCNN
- Category-specific and open-vocabulary: Grounding-DINO

Need to know categories to be detected

Kirillov et al., "Segment Anything," ICCV, 2023
He et al., "Mask R-CNN," ICCV, 2017.
Liu et al., "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," arXiv, 2023.

Image to 3D Models

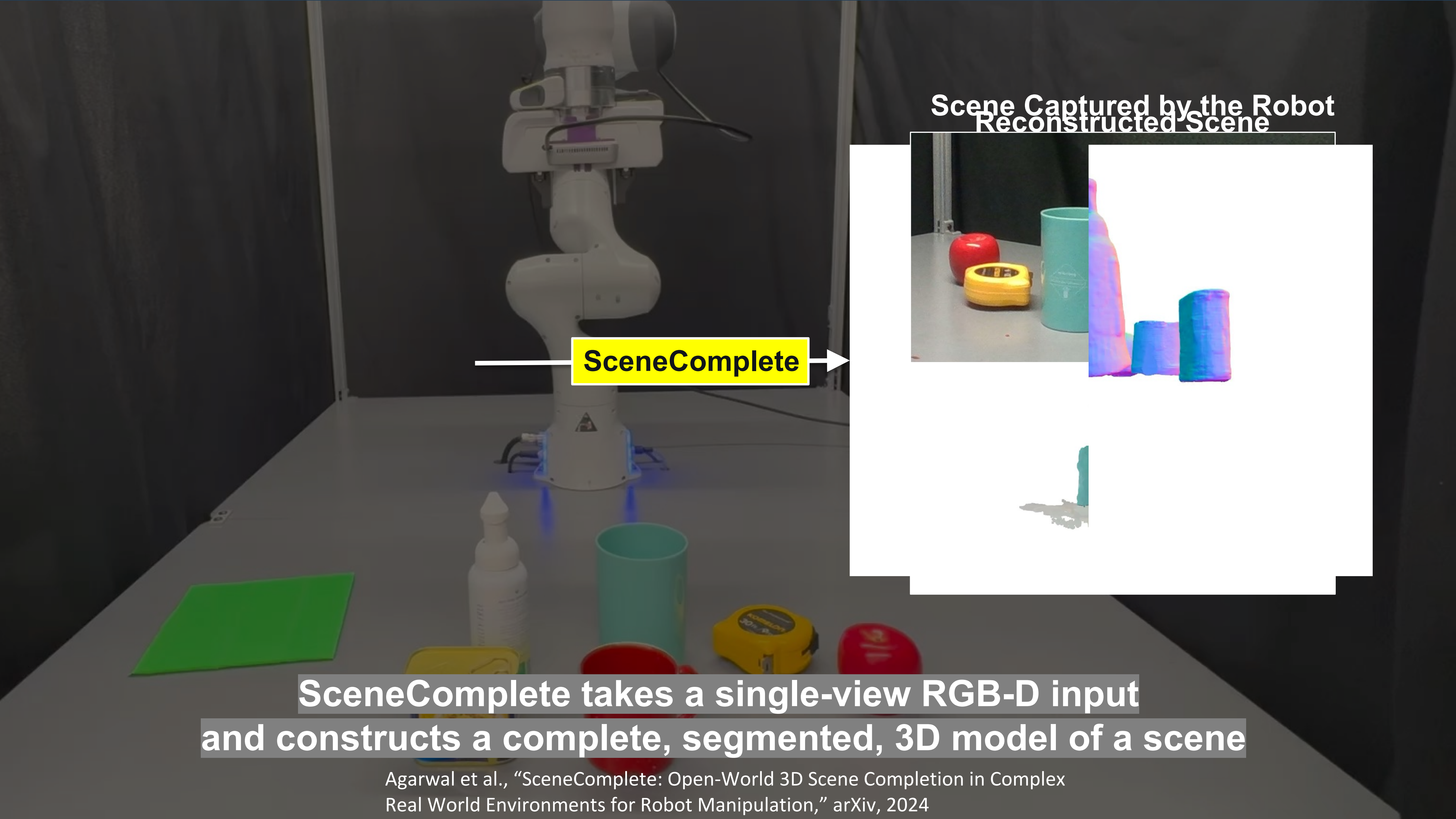
- Many existing models: RGB -> 3D
- Zero-1-to-3, InstantMesh, Instant3D

- Caveat: Usually they don’t work well with partial object images (need inpainting)
- Many methods work better if we know the name of the object

Liu et al., “Zero-1-to-3: Zero-shot One Image to 3D Object,” ICCV, 2023.
Xu et al., “InstantMesh: Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models,” arXiv, 2024.
Li et al., “Instant3D: Fast Text-to-3D with Sparse-View Generation and Large Reconstruction Model,” arXiv, 2023.

3D Back-Projection

- Shape completion methods usually only work with RGB images
- So they don’t know the actual “size” of the 3D shape
- After obtaining the mesh for an object, we need to back-project it
- Keyword: pointcloud registration



SceneComplete takes a single-view RGB-D input and constructs a complete, segmented, 3D model of a scene

Agarwal et al., "SceneComplete: Open-World 3D Scene Completion in Complex Real World Environments for Robot Manipulation," arXiv, 2024

Object Tracking

- While the object is being moved, we need to keep track of it!
- Otherwise we won't know object correspondences across states

- Three commonly used tracking modules:
- Mask tracker: Segment-Anything 2

Ravi et al., "SAM 2: Segment Anything in Images and Videos," ICLR, 2025.
Doersch et al., "TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement," arXiv, 2023.
Karaev et al., "CoTracker: It is Better to Track Together," ECCV, 2024.
Wen et al., "FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects," CVPR, 2024.

Object Tracking

- While the object is being moved, we need to keep track of it!
- Otherwise we won't know object correspondences across states

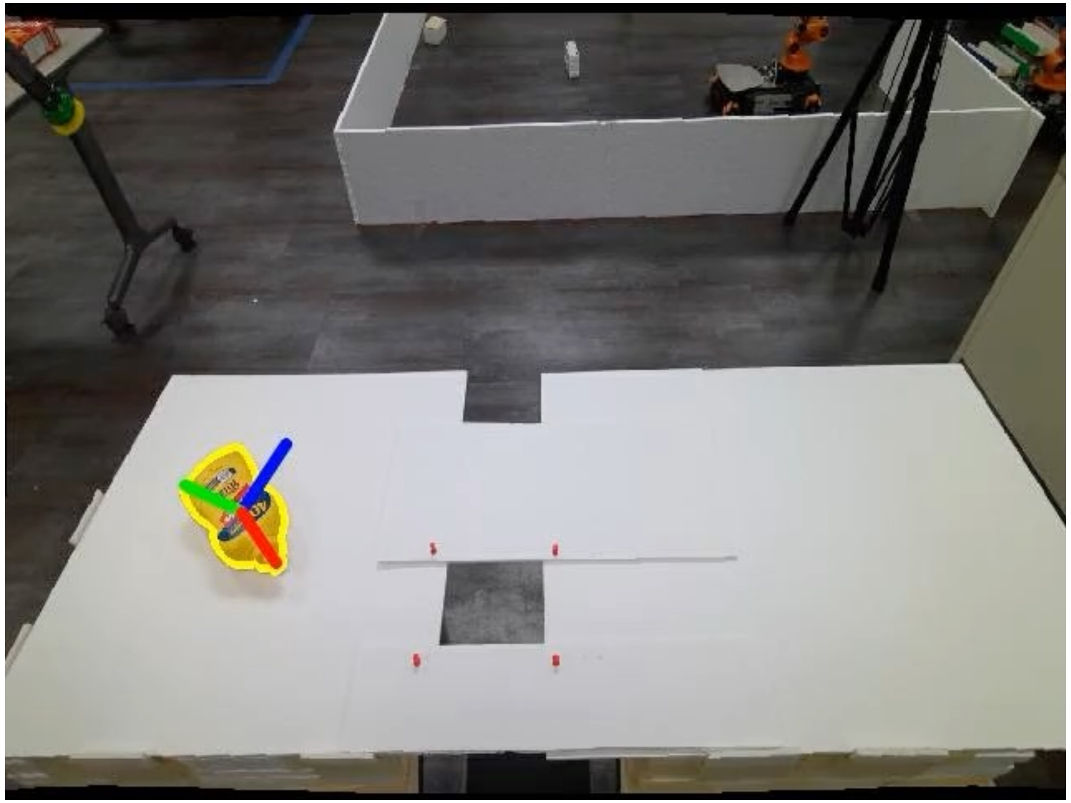
- Three commonly used tracking modules:
- Mask tracker: Segment-Anything 2
- Point tracker: Track-Any-Point, CoTracker2

Ravi et al., "SAM 2: Segment Anything in Images and Videos," ICLR, 2025.
Doersch et al., "TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement," arXiv, 2023.
Karaev et al., "CoTracker: It is Better to Track Together," ECCV, 2024.
Wen et al., "FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects," CVPR, 2024.

Object Tracking

- While the object is being moved, we need to keep track of it!
- Otherwise we won't know object correspondences across states

- Three commonly used tracking modules:
- Mask tracker: Segment-Anything 2
- Point tracker: Track-Any-Point, CoTracker2
- Pose tracker: Foundation Pose

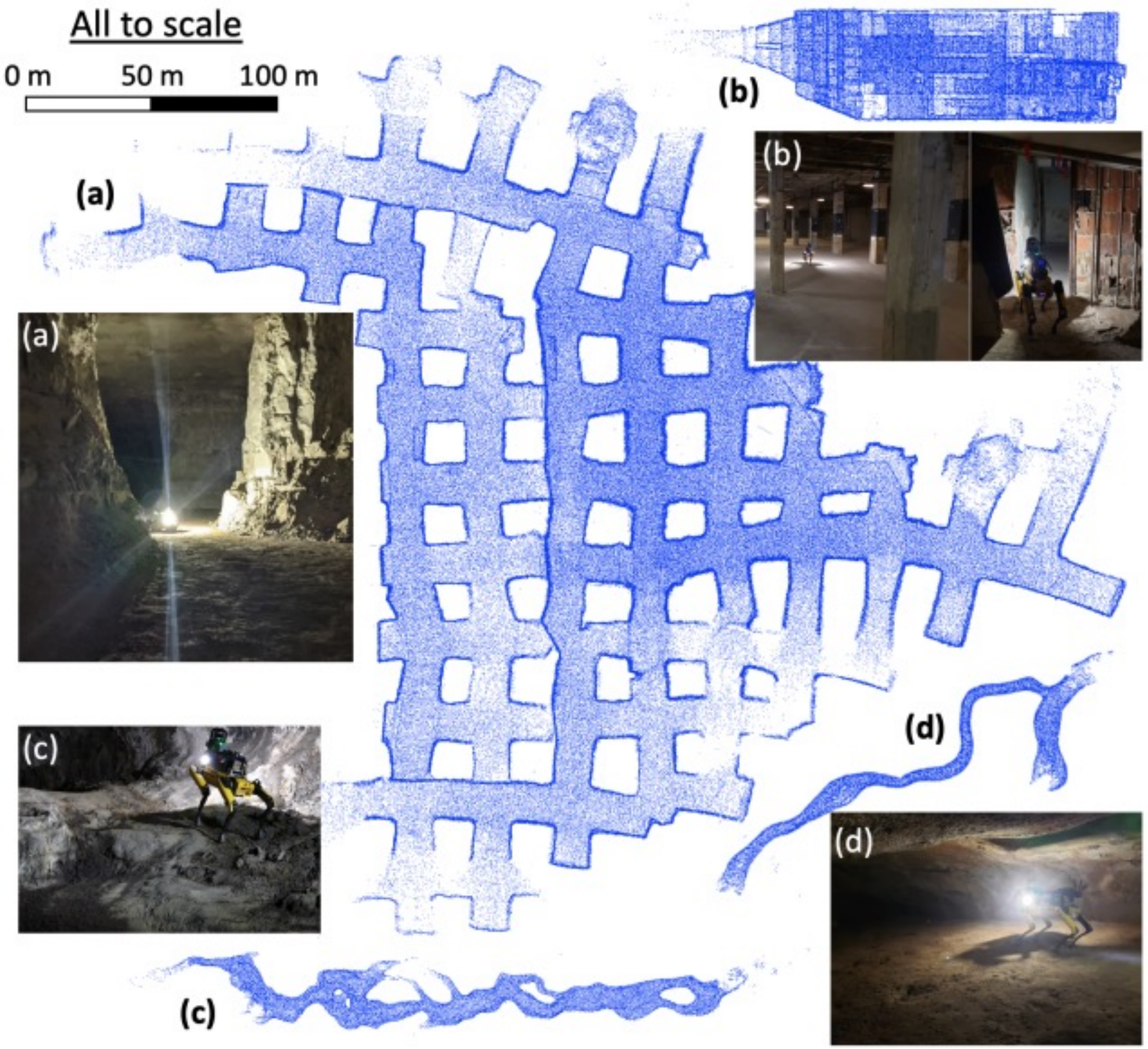


Ravi et al., "SAM 2: Segment Anything in Images and Videos," ICLR, 2025.
Doersch et al., "TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement," arXiv, 2023.
Karaev et al., "CoTracker: It is Better to Track Together," ECCV, 2024.
Wen et al., "FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects," CVPR, 2024.

Summary

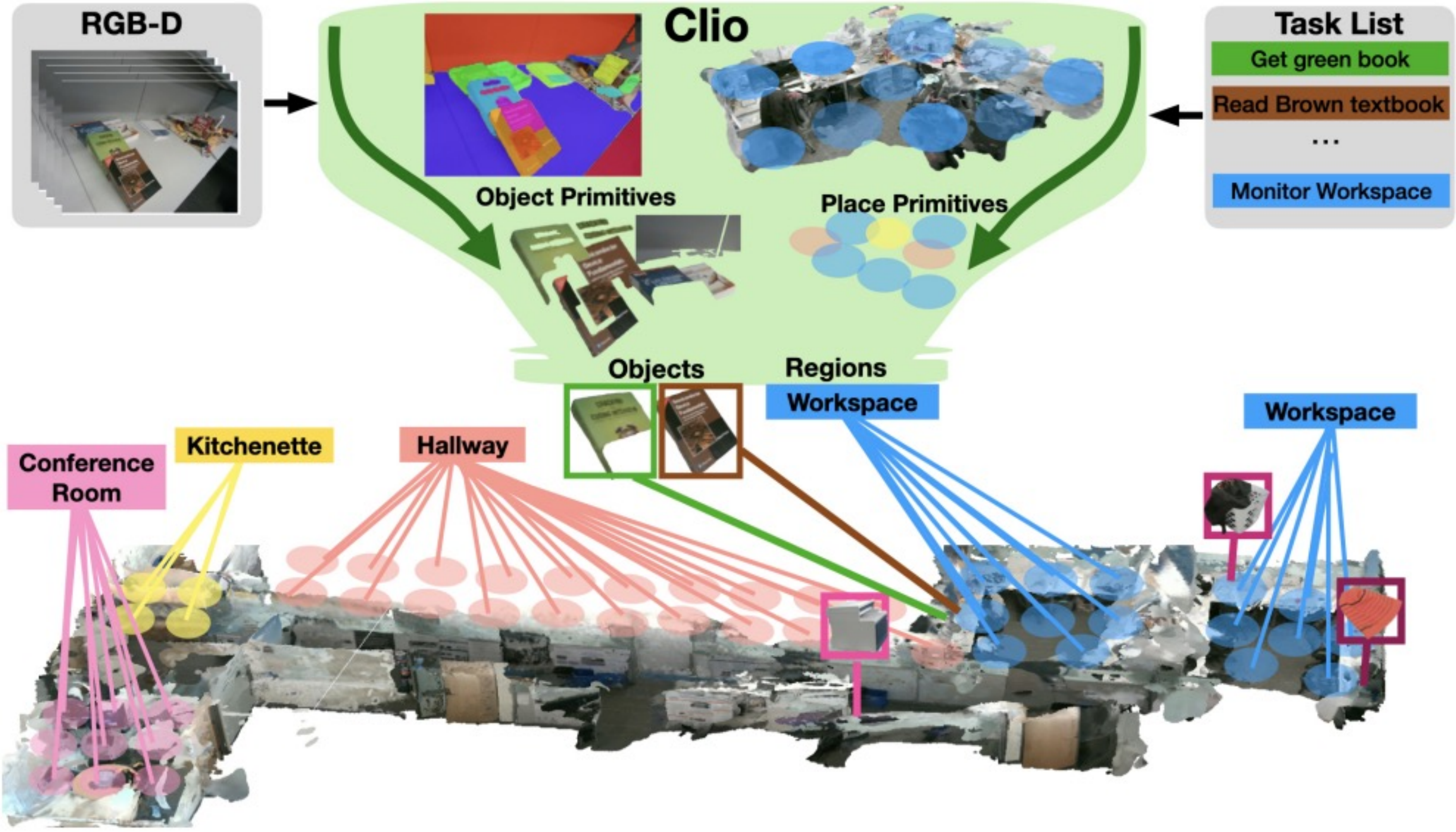
- Many 2D and 3D computer vision techniques are needed to build an object-centric state representation
- Now we have better and better foundation models for ALL of them
- However, we still don't have a "single" foundation model for all tasks
- Moreover, many models are not tuned for robotics purposes
- Different planning and control algorithms may need different levels of details

Advanced: Spatial Localization and Mapping



Reinke et al., "LOCUS 2.0: Robust and Computationally Efficient Lidar Odometry for Real-Time 3D Mapping," R-AL, 2022.

Advanced: Object-Centric SLAM

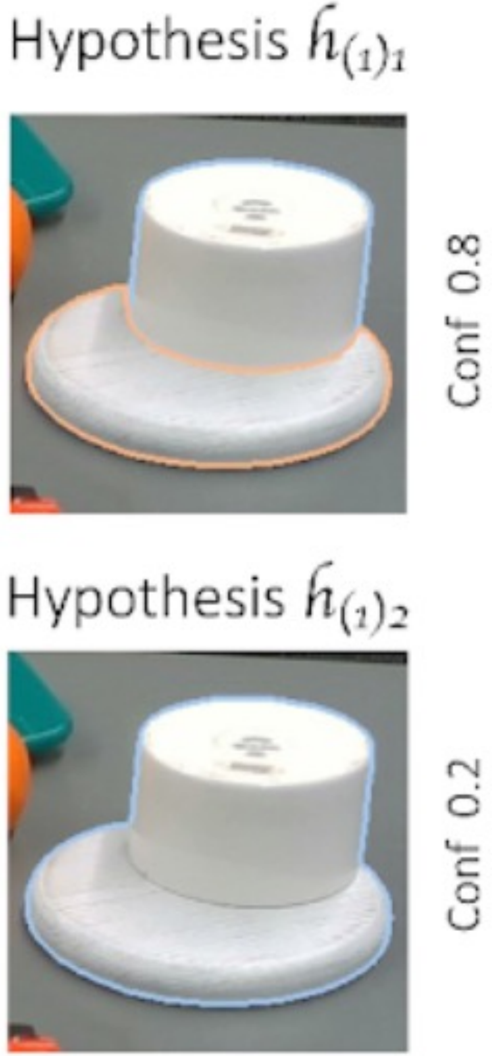


Maggio et al., "Clio: Real-time Task-Driven Open-Set 3D Scene Graphs," R-AL, 2024.

Advanced: Segmentation Under-Specification

- Depending on the task, you need to segment objects at different granularities

Advanced: Segmentation Uncertainty



- Interaction is usually needed to dis-ambiguate

Fang et al., “Embodied Uncertainty-Aware Object Segmentation,” IROS, 2024.

Many Other Frontier Topics in Perception

- Depth sensor denoising
- Articulated object perception
- Active sensing of physical properties
- SLAM with dynamic objects
- Task-driven representation of uncertainty

From States to Actions: The Hierarchy

- Most systems involve a two-level design: high-level and low-level

Low-Level Action Interface

- **Lowest-Level Action:** how much current should I apply?
- Usually run at >1000Hz

- **“Low-Level” Action:**
    - target position / velocity for the robot joints
    - target position / velocity for the robot end-effector

Low-Level Action Interface

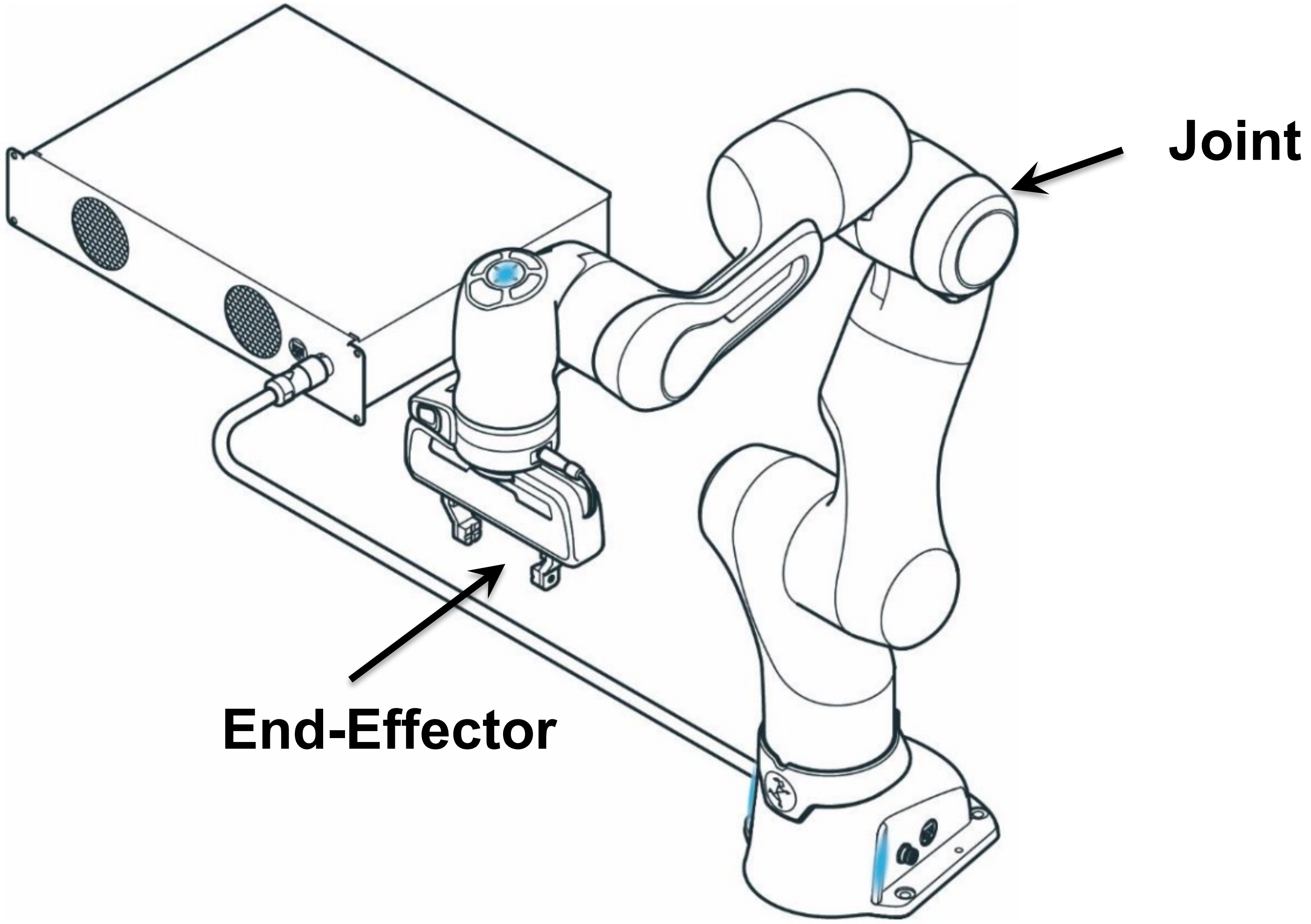


Figure from Franka Research 3 Manual

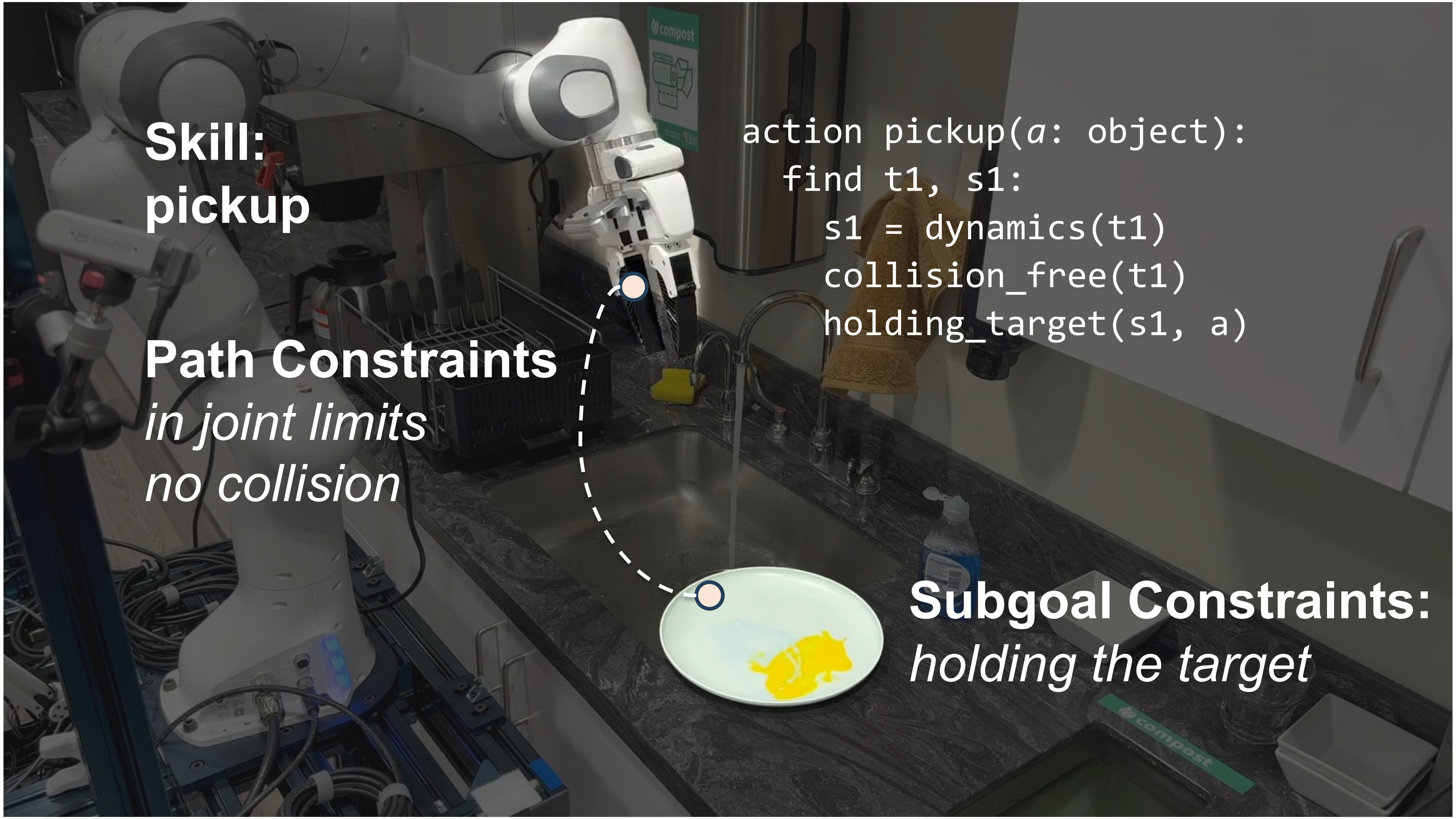
High-Level Action Interface

- **High-Level Actions** are usually object-centric
- Different algorithms may use different granularities

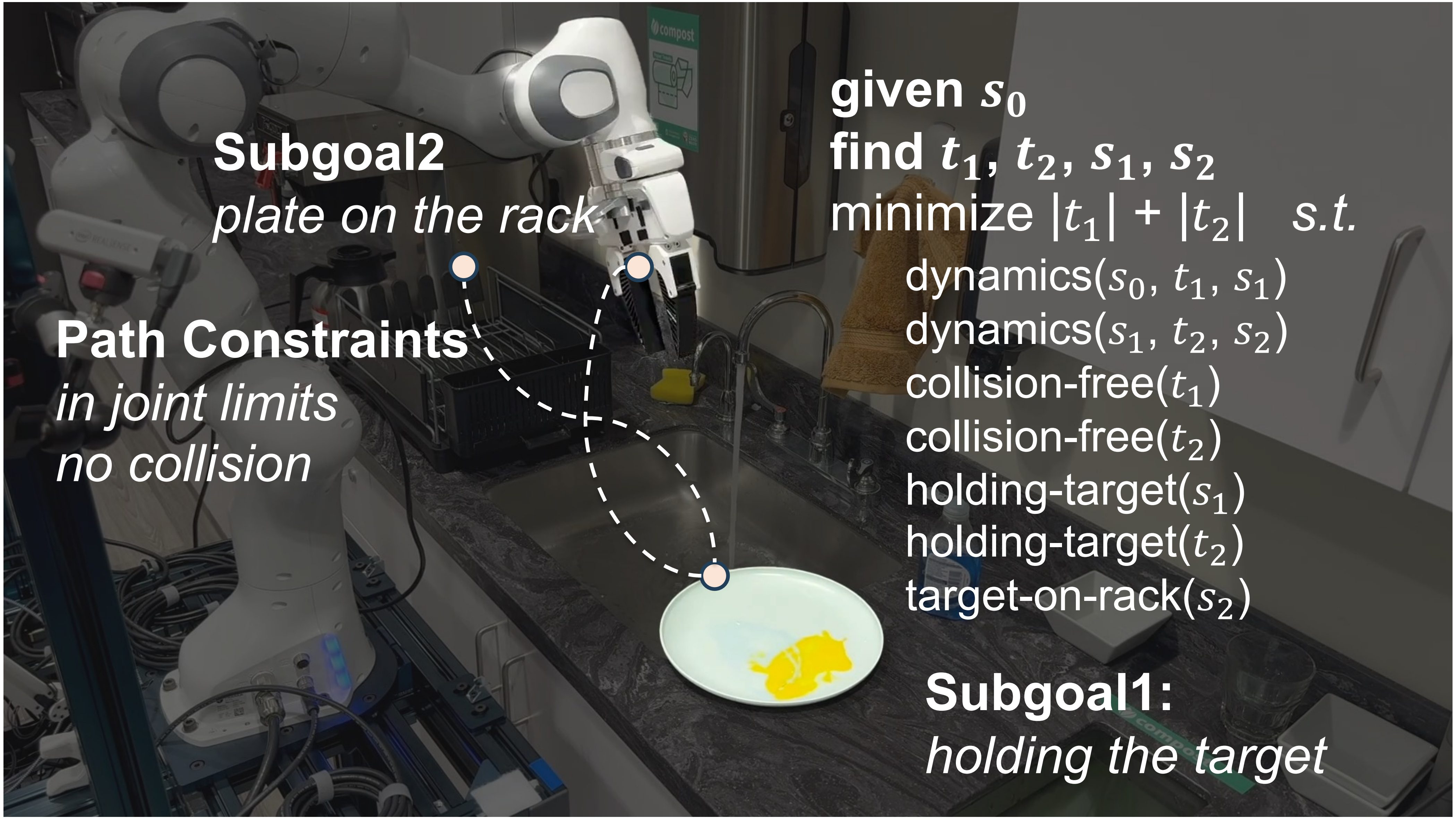
```
action grasp(object):
  grasp_pos = find_grasp(object)
  traj = find_trajectory(current_pos(), grasp_pos)
  execute(traj)
  close_gripper()


action place(object, surface):
  place_pos = find_place(object, surface)
  traj = find_trajectory(current_pos(), place_pos)
  execute(traj)
  open_gripper()
```

Integrated Low-Level and High-Level Actions



Integrated Low-Level and High-Level Actions



Summary

- **Low-level action:** joint and end-effector commands
- **High-level action :** object-centric commands
- **Integrated low-level and high-level action:** usually based on constrained optimization frameworks

Advanced: Execution Monitoring

- How to react to human perturbation and other endogenous events in a multi-level system?
- Simple solution, but usually not scalable: perform action selection at all layers at a high frequency