Design Choices in Modern Embodied AI Environments

Building and Working in Environments for Embodied AI (part III)

CVPR 2022 Tutorial

UC San Diego

清華大學
Tsinghua University

SFU SIMON FRASER UNIVERSITY

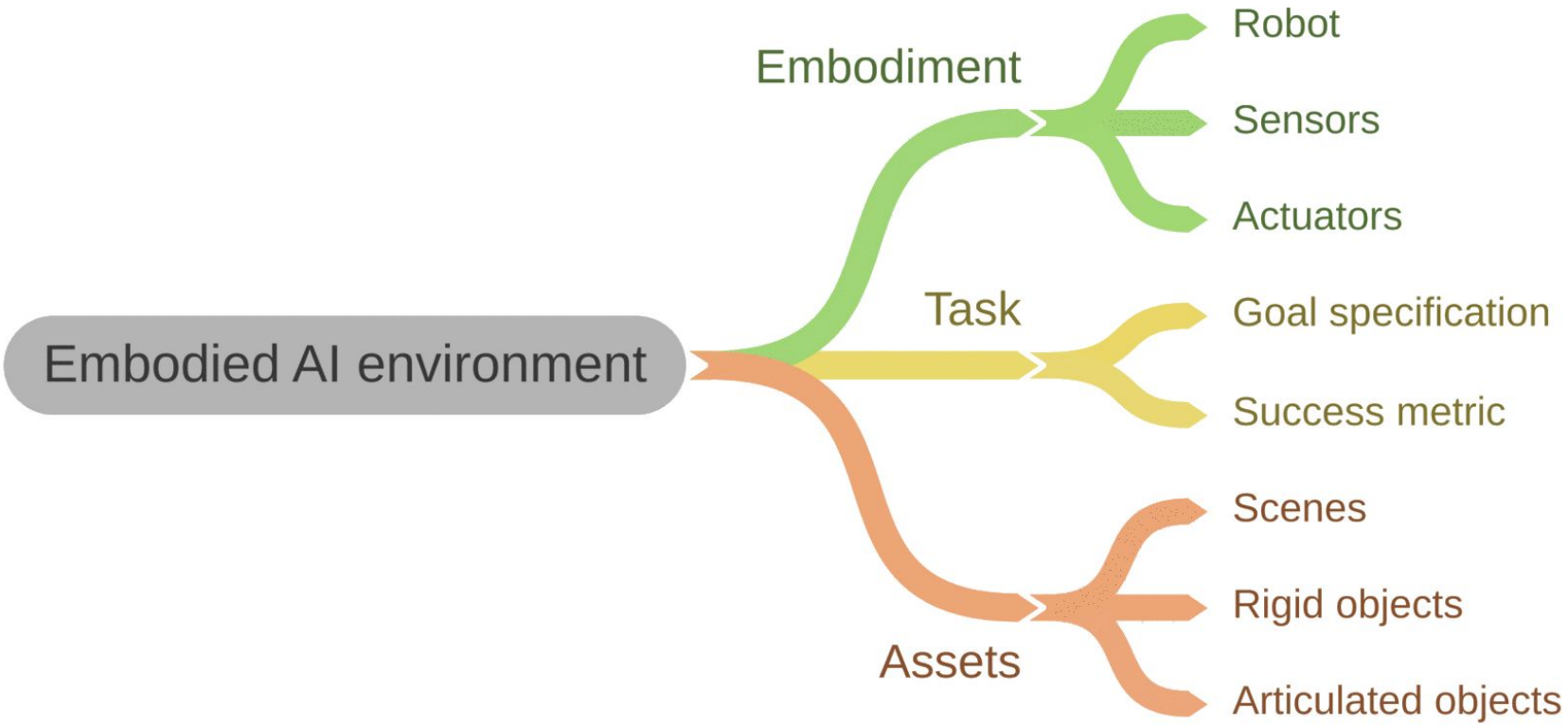
Overview

- We are talking about:
  - How to **design embodiment**?
  - How to **define a task**?
  - What kind of **assets** are needed?
- We will show several case studies of existing Embodied AI environments

Overview

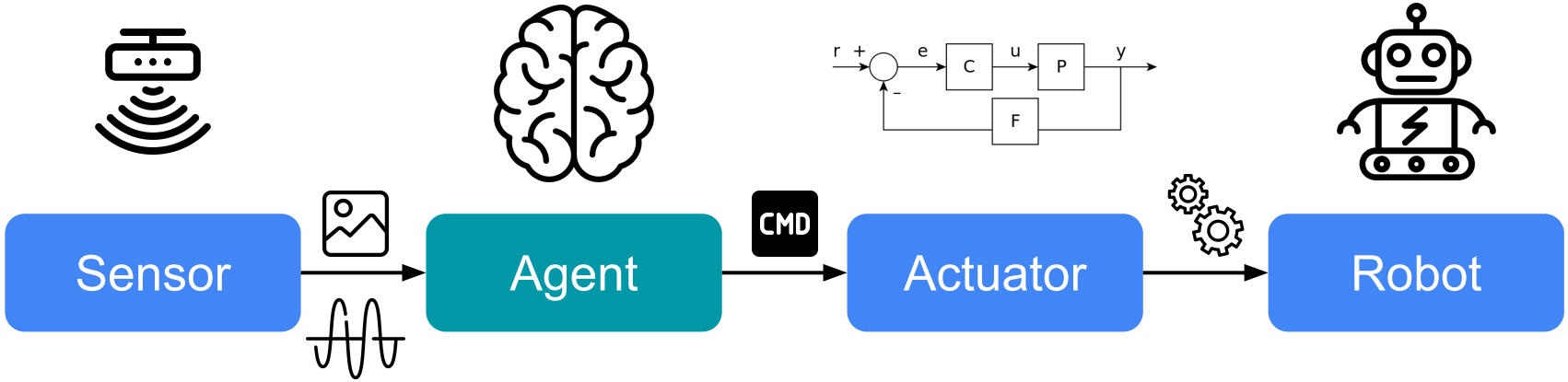
- This section is intended for who want to:
  - Find a suitable environment to start their projects
  - Have a deeper look at environments they are using
  - Customize or create a new environment
- All design choices are discussed in the simulation context

Most Common Design Choices



Design Choice: Embodiment

- Robot hardware
- Sensors
- Actuators



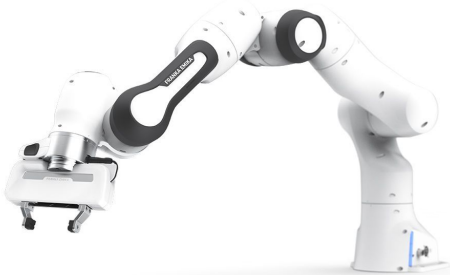
Figures from Wikipedia and flaticon

Embodiment: Robot Hardware

- Physical body with various sensorimotor capacities
- Shape the cognition and action

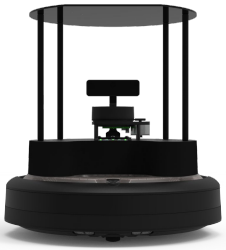
Legged robot

Unitree A1 Robot



Manipulator

Franka Emika Panda



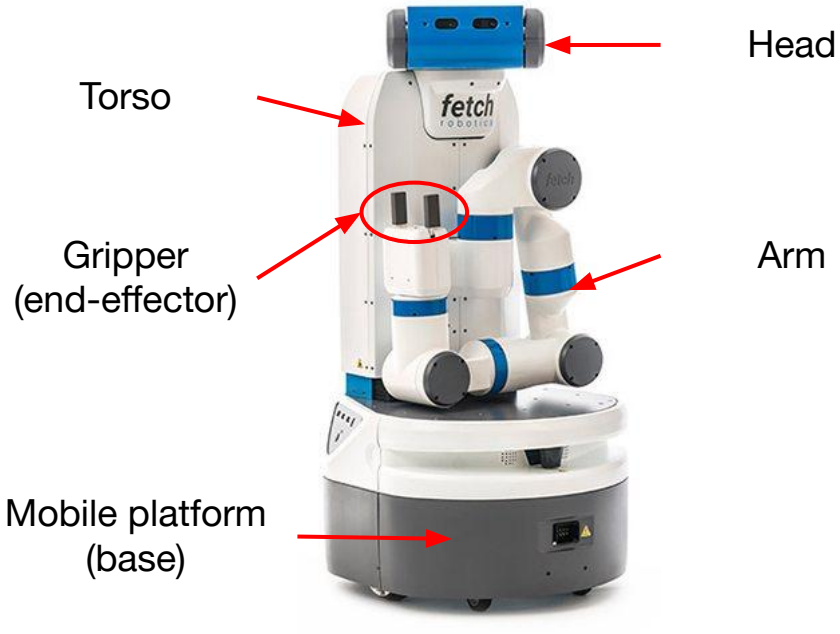
Wheeled robot

TurtleBot

Mobile manipulator

Fetch

Common Robot Components



Embodiment: Sensor

- Common types of sensors
  - Visual perception
  - Robot proprioception
  - Haptic/tactile perception
  - GPS and compass (IMU)

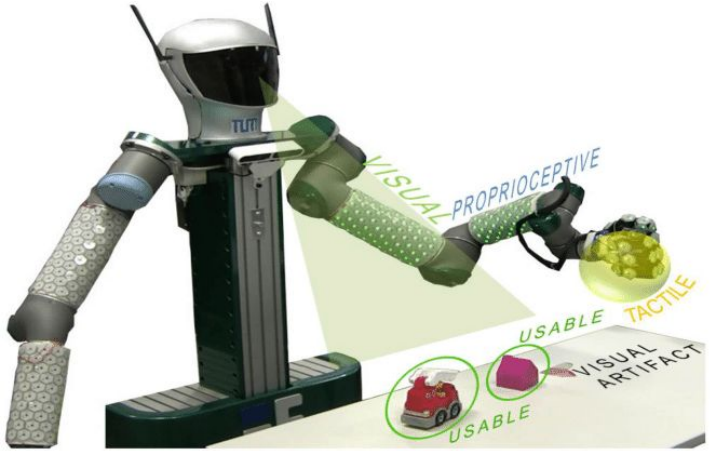
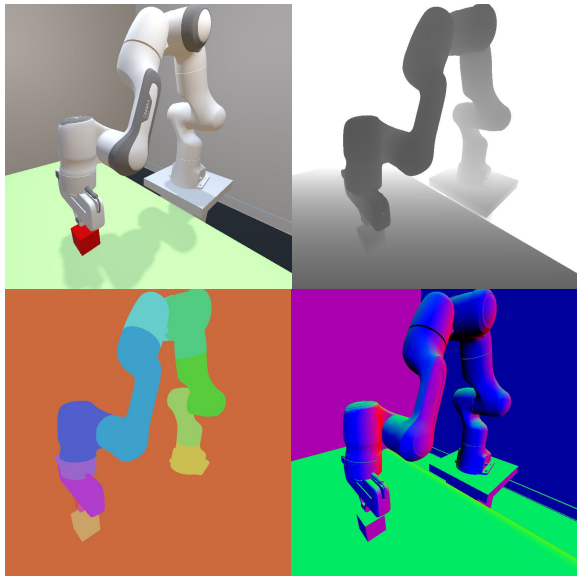


Figure from Yielding Self-Perception in Robots Through Sensorimotor Contingencies

Sensor: Visual Perception

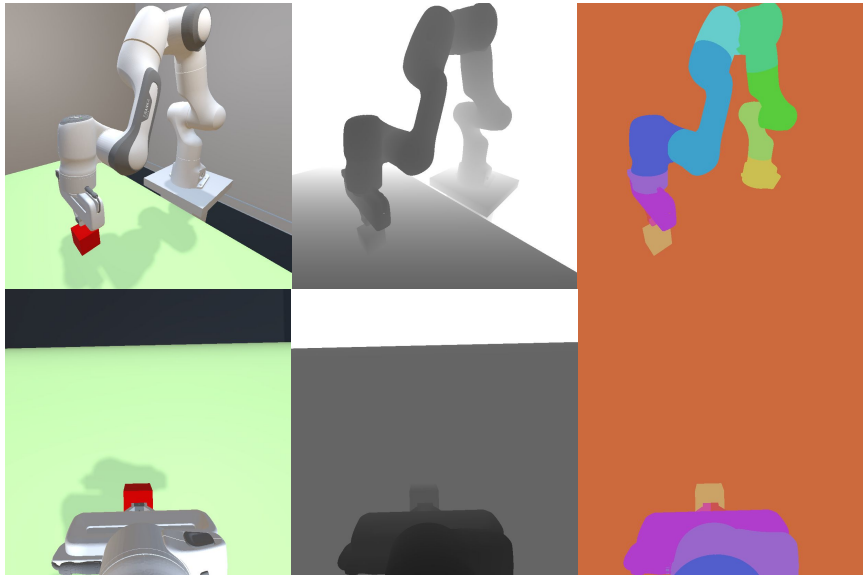
- Common signal types
  - RGB
  - Depth
  - Semantic/instance masks
- Other signal types
  - Normal
  - Optical flow
  - …



ManiSkill 2022: rgb, depth, semantic, normal

Sensor: Visual Perception

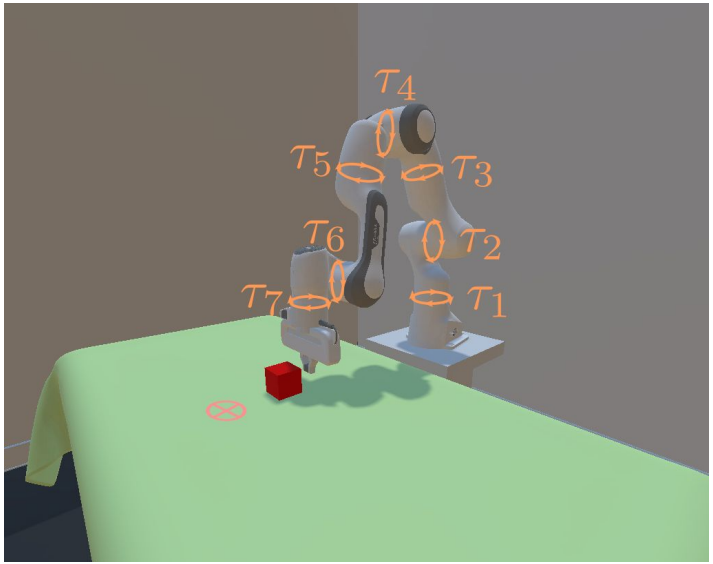
- Common placement
  - Head
  - Arm
  - Third-view



Top: third-view; bottom: arm (wrist)

Sensor: Robot Proprioception

- Joint state
  - Position
  - Velocity
  - Torque
- Link state
  - End-effector pose



Sensors: Haptic/Tactile Perception

- Whether the gripper is grasping something
- Force-torque sensor: measuring external force and torque

Sensors: GPS and Compass

- Odometry: GPS (relative position) and Compass (relative orientation)
- Used to localize the agent relative to its initial pose

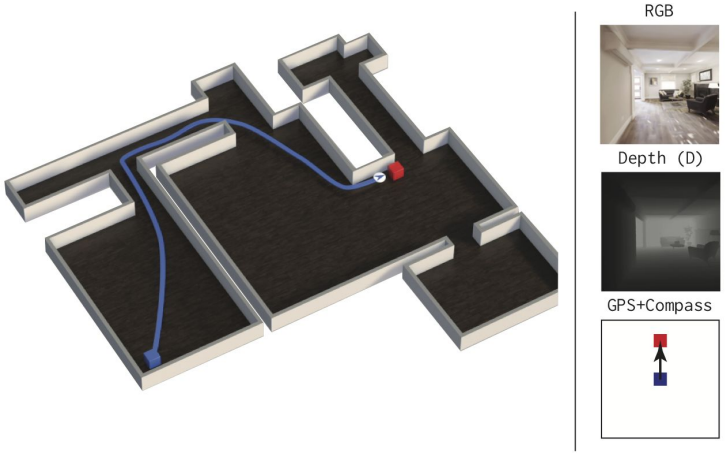
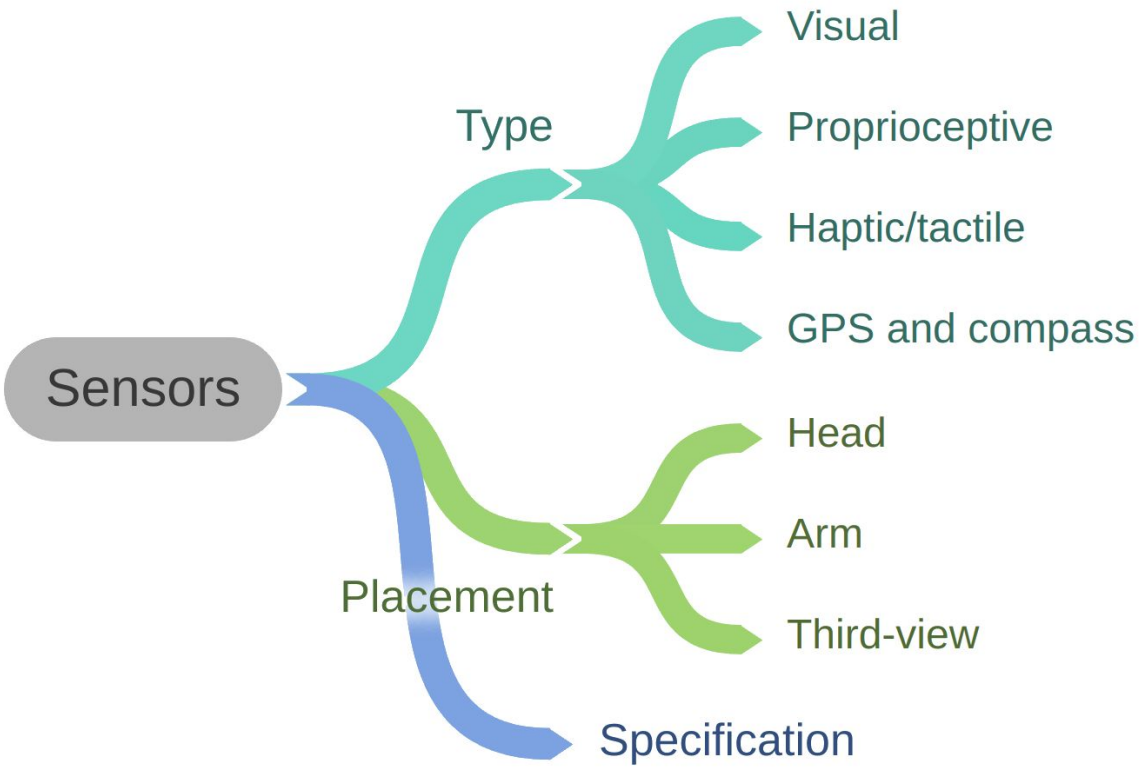


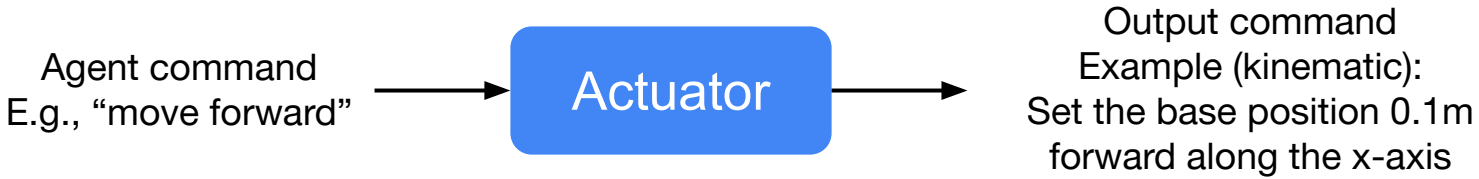
Figure from DD-PPO

Design Choices for Sensors



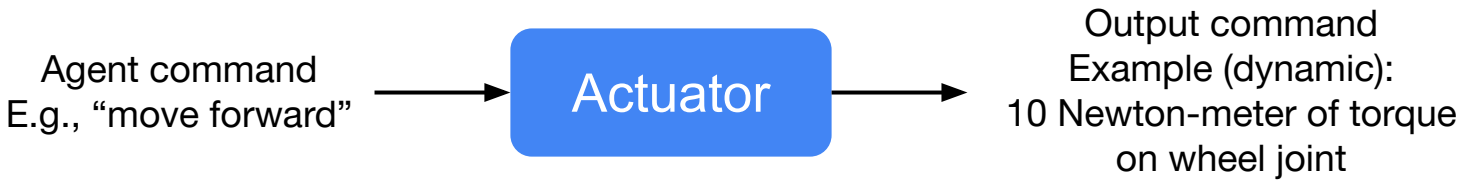
Embodiment: Actuator

- *“An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system”* from Wikipedia



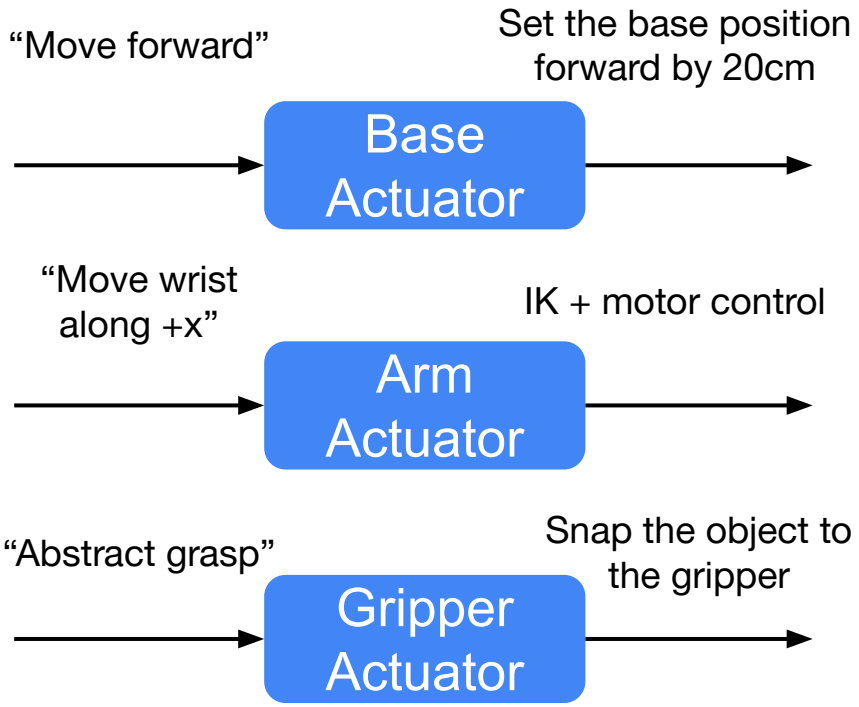
Embodiment: Actuator

- *“An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system”* from Wikipedia



Actuator: Example

ArmPointNav



Design Choice: Task Specification

- Goal specification
- Success metric

Task: Goal Specification

- Geometric position
- Object category
- Semantic/instance masks
- Language instruction
- Goal image
- Successful demonstration
- …

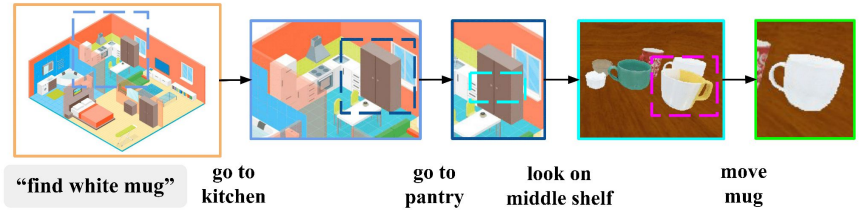
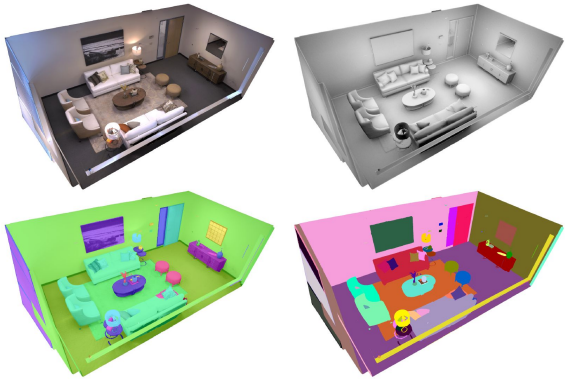


Figure from Multi-Layer Semantic and Geometric Modeling with Neural Message Passing in 3D Scene Graphs for Hierarchical Mechanical Search

Task: Success Metric

- Common success metrics
  - Navigation: The agent stops close enough to the goal
  - Manipulation: The pose of the target object is close enough to the goal
- Additional success requirements
  - The arm returns to a resting position
  - The robot is static
  - The states of target objects satisfy success predicates
  - …

Design Choice: Assets



Non-interactive scenes

[Replica Dataset]

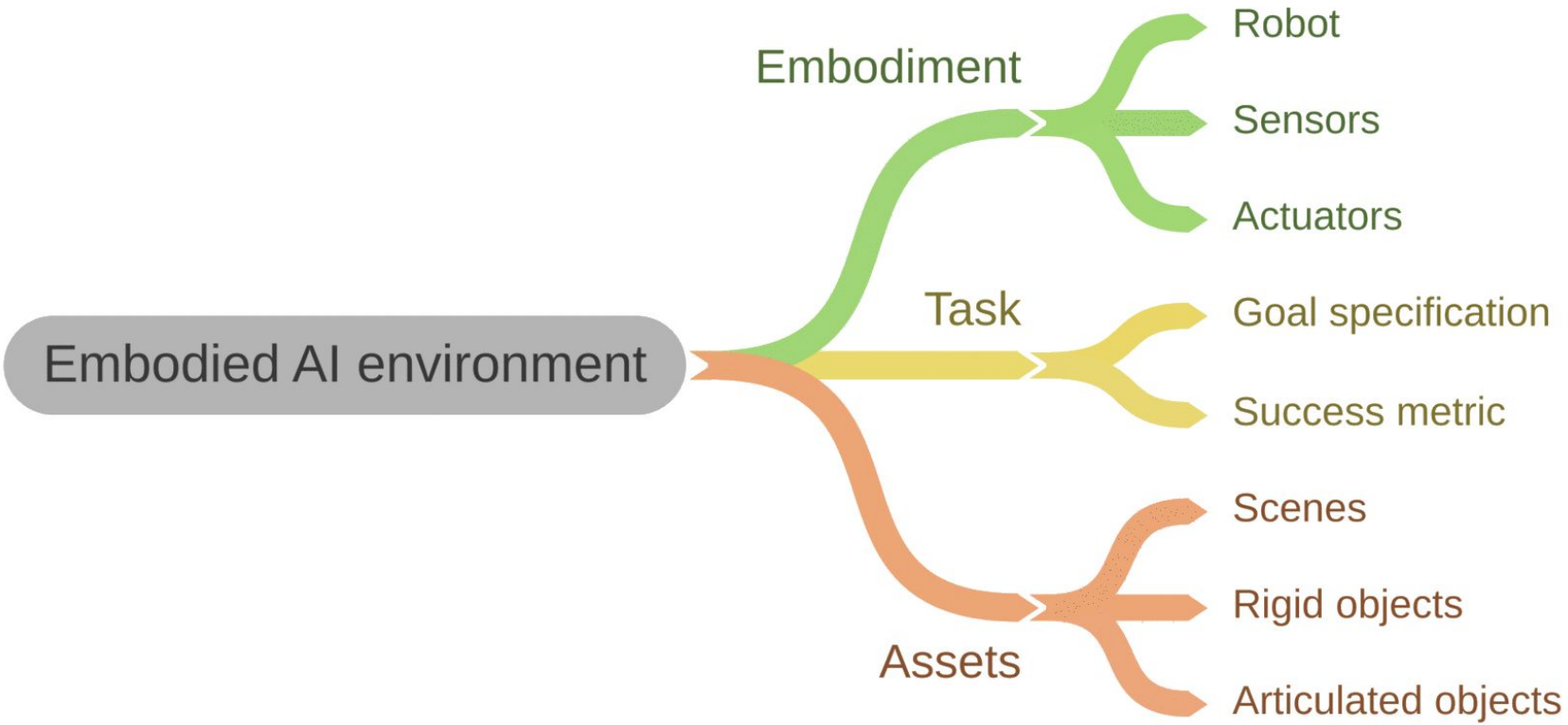
Rigid objects

[YCB Object Set]

Articulated objects

[PartNet-Mobility Dataset]

Recap: Design Choices



Physical vs. Non-physical Simulation

- Certain design choices do not need physical simulation
  - Sensor: (perfect) semantic/instance mask
  - Actuator: high-level input command and kinematic output command, e.g., like “pick an apple” implemented by setting the position of apple without simulation
  - Assets: non-interactive scenes

Case Studies:
AI Habitat, AI2THOR, ManiSkill

The cases studied here are not exhaustive. There are also other embodied AI environments, like BEHAVIOR (iGibson), MetaWorld, Robosuite, RLBench, ThreeDWorld, etc.

Case Study I: AI Habitat

- Habitat 1.0: Navigation
  - PointNav
  - ObjectNav
  - Instruction following
  - Embodied QA

The figures and videos in this section are from Habitat unless specified

Case Study I: AI Habitat

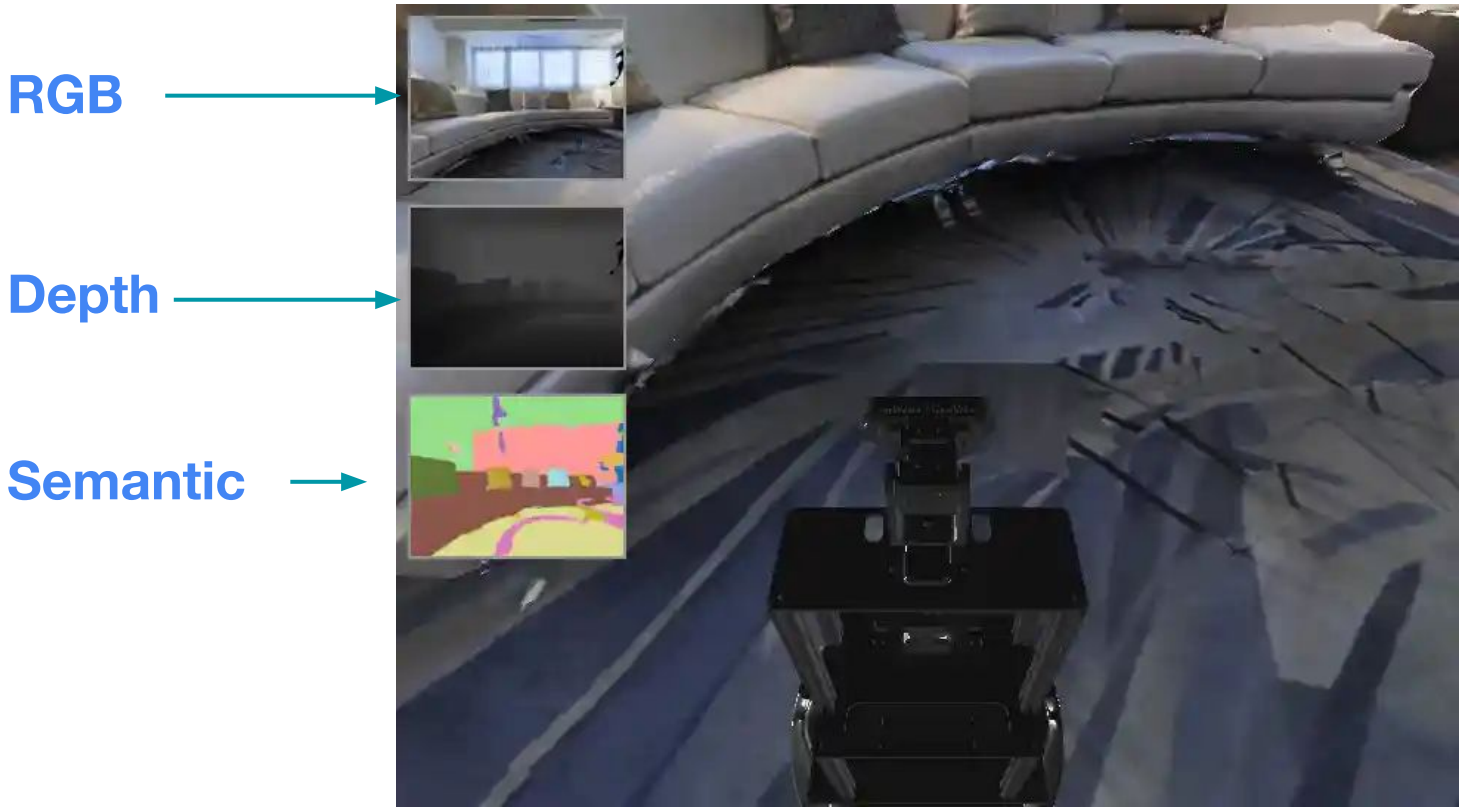
- Habitat 2.0: Mobile manipulation
  - Home Assistant Benchmark
  - Under active development
- We focus on Habitat 1.0 in this talk

Video from https://sites.google.com/view/hab-m3

Embodiment

- Robot: a cylindrical primitive shape with a diameter of 0.2m and a height of 1.5m
- Visual sensors
  - RGB-D camera at a height of 1.5m and oriented to face ‘forward’
  - Semantic instance masks (optional, for training)
- Proprioceptive sensors
  - Perfect GPS and compass (to compute relative goal position)

Embodiment: Visual Sensors



The robot is only for visualization

Embodiment: Actuator

- Input: a discrete action in *move forward* (0.25m), *turn left* (10 deg), *turn right* (10 deg) and stop
- Output: set the pose of the navigation agent without physical simulation
- Navigation constraints and collision response are based on **NavMesh** (explained in the next slide)

NavMesh

- A navigation mesh (NavMesh) is a collection of two-dimensional convex polygons (i.e., a polygon mesh) that define which areas of an environment are traversable by an agent with a particular embodiment.

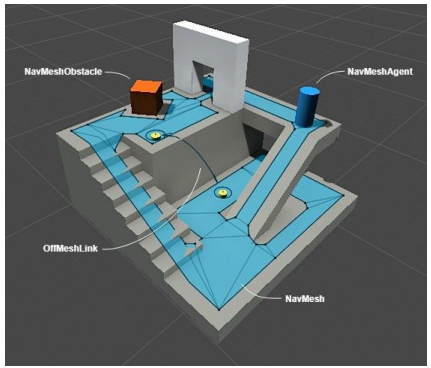
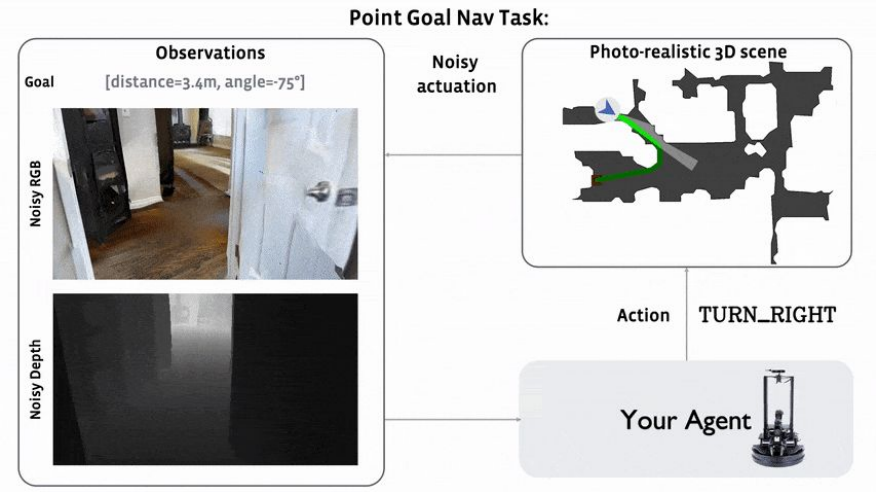


Figure from Dave Johnston's blog

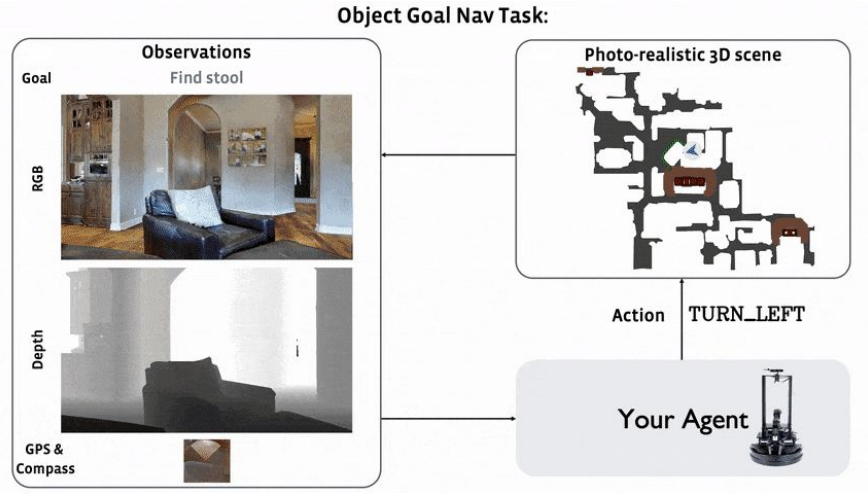
Task Specification

- PointNav
  - Geometric goal: a 2D position relative to the agent's start location
  - *An episode is successful if on calling the STOP action, the agent is within 0.36m (2x agent-radius) of the goal position.*



Task Specification

- ObjectNav
  - Semantic goal: a category
  - *An episode is successful if on calling the STOP action, the agent is within 1.0m from any instance of the target object category AND the object can be viewed by an oracle*



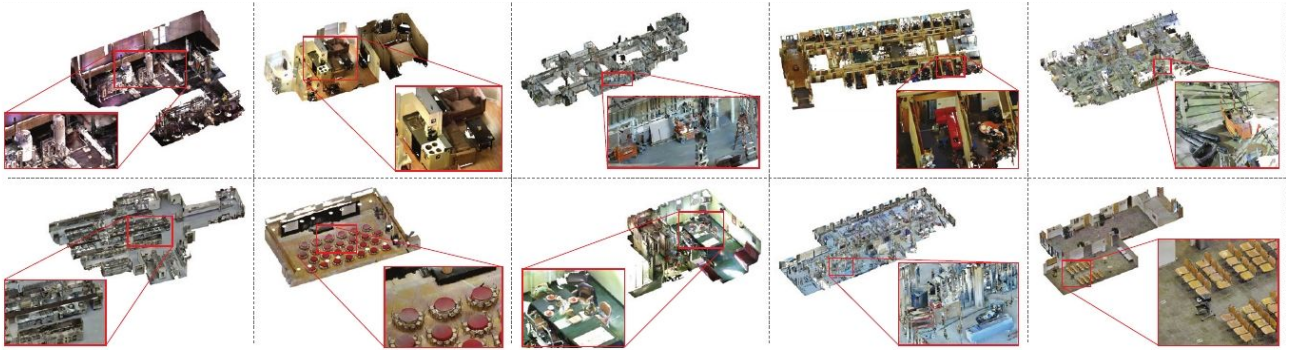
Task Specification: Metrics

- Success weighted by Path Length (SPL)
  - Measure the efficiency to reach the goal

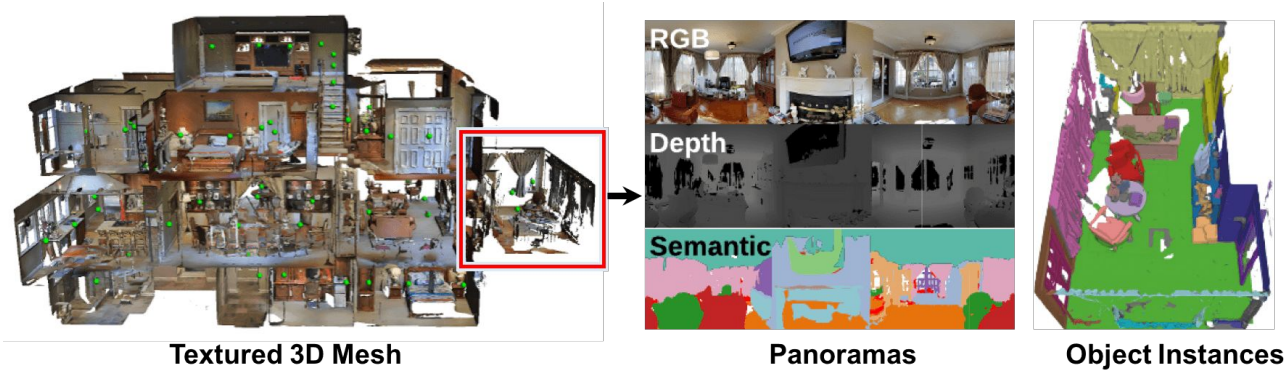
$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

$$\begin{aligned} \text{where, } l_i &= \text{length of shortest path between goal and target for an episode} \\ p_i &= \text{length of path taken by agent in an episode} \\ S_i &= \text{binary indicator of success in episode } i \end{aligned}$$

Assets: Scenes



Gibson



Textured 3D Mesh | Panoramas | Object Instances

Matterport3D

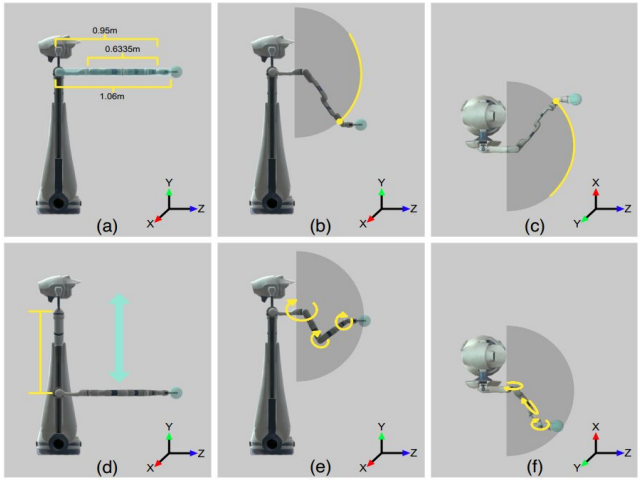
Case Study II: AI2THOR

- iTHOR:
  - Navigation and high-level object interaction
  - Instruction following
  - Embodied QA
- ManipulaTHOR:
  - Mobile manipulation (pick-and-place)

The figures and videos in this section are from AI2THOR unless specified

Embodiment: Robot

- Arm based on Kinova Gen3
- The arm height is adjustable



Embodiment: Visual Perception

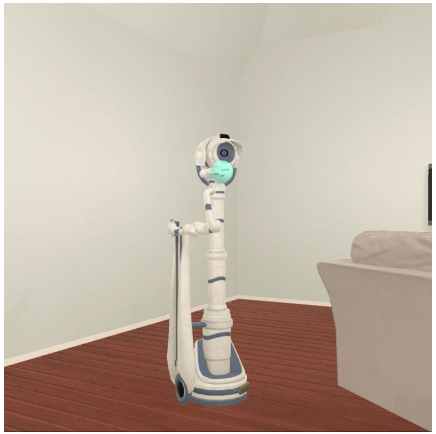
- Types of sensor signals
  - RGB-D
  - Instance masks and 3D bounding boxes (optional for training)
- Camera placement
  - Mounted on the head, adjustable (height and angle)

Embodiment: Other Sensors

- iTHOR
  - Base position and rotation (a.k.a. GPS and compass)
  - Primitive action binary feedback (open, pick, slice, etc.)
- ManipulaTHOR
  - Base position and rotation (a.k.a. GPS and compass)
  - End-effector (or joint) poses
  - Grasp binary feedback (based on abstract grasp)

Embodiment: Actuator

- Base
  - Discrete actions (RotateLeft, RotateRight, MoveForward, etc.)
- Arm (ManipulaTHOR)
  - Discrete actions (MoveArmX, MoveArmY, MoveArmHeight, etc.)
  - Target end-effector pose

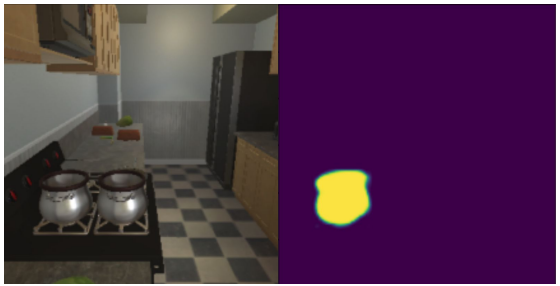


The arm movement

Embodiment: Actuator

- Head
  - Adjust the height and pitch angle of the camera
- Gripper
  - Mask-based primitive actions (for iTHOR)
  - Magnet-based abstract grasp (for ManipulaTHOR)

The head tilts to adjust its camera



Specify target object by seg. mask (iTHOR)

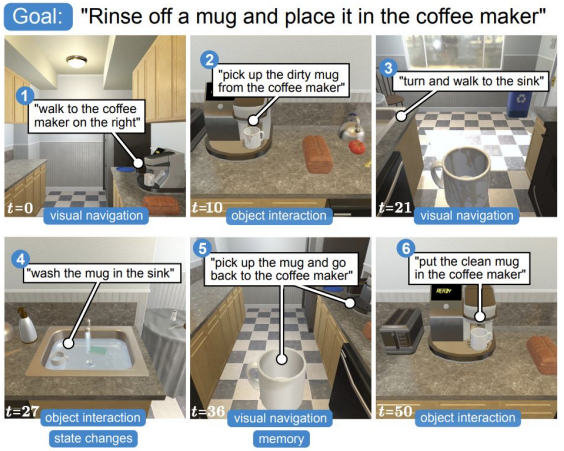
Assets

- Scenes
  - 120 room-scale scenes designed by artists
  - kitchens, living rooms, bedrooms, and bathrooms
- Objects
  - Over 100 object types
    - Rigid objects such as knife
    - Articulated objects such as fridge
  - State changes
    - Temperature, is broken, is dirty, etc.

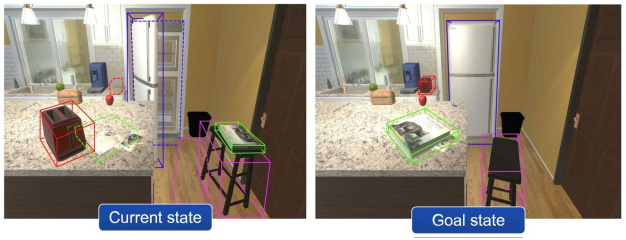
Scenes from AI2-THOR

Task: Goal Specification

- Semantic
  - Target object category (RoboTHOR Challenge)
- Language instructions
  - High-level or step-by-step human instructions (ALFRED Challenge)
- Visual observations (Rearrangement Challenge)



Example of ALFRED Challenge



Example of Rearrangement Challenge

Success Metrics (iTHOR)

- How to define success?
  - Check the difference between current and target object poses
  - Check the states of target objects (e.g., is food cooked?)
- Metrics
  - Success rate
  - Success rate weighted by path length (SPL)

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \cdot \frac{\ell_i}{\max(p_i, \ell_i)}$$

Task in ManipulaTHOR

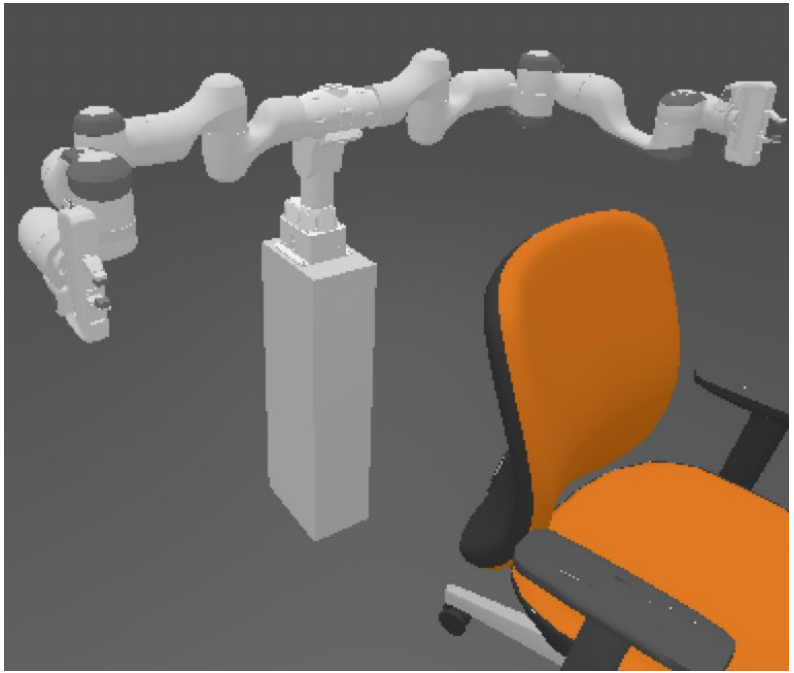
- Supported task: ArmPointNav
- Geometric goal: 3D goal position of the target object
- How to define success?
  - Check the difference between current and target object poses
- Additional metrics
  - Success rate without disturbance/collision

Case Study III: ManiSkill

- Physical manipulation + object-level generalization

Embodiment: Robot

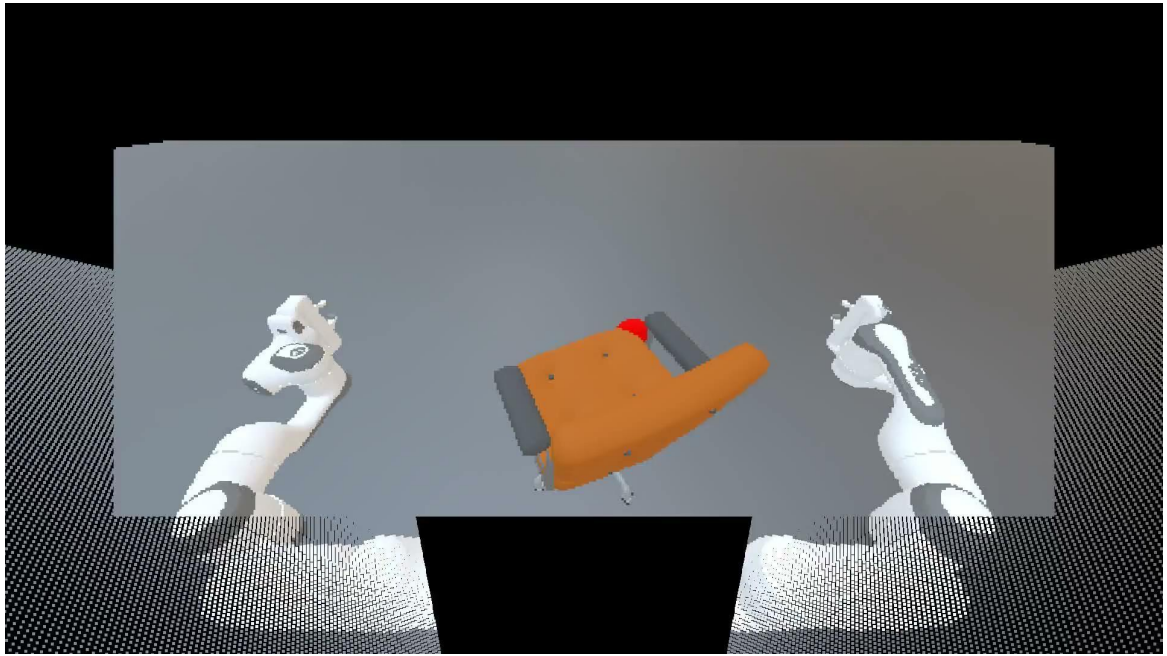
- Mobile platform
  - 3 DoF: x-y-plane translation and z-axis rotation
- Sciurus torso
  - 1 DoF (adjust height)
- Franka Panda arm(s)
  - 7 DoF arm
  - 2 DoF gripper



Embodiment: Sensors

- Visual sensors
  - RGB-D and semantic/instance masks
  - 3 cameras mounted on the top of the robot, with a 120° FOV per camera to provide a panoramic view
- Other sensors
  - Joint positions and velocities
  - Perfect GPS and compass

Embodiment: Visual Sensors



Fused Point Cloud

Embodiment: Actuator

- Input: target velocities of joints
- Output: torques on joints to achieve the target velocities
- It is equivalent to PD joint velocity controller in robotics
- Full physical simulation without abstraction (including grasp)

Task Specification

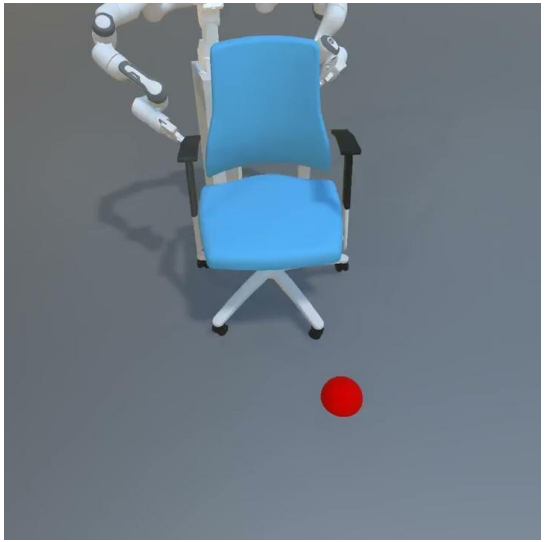
OpenCabinetDoor

OpenCabinetDrawer

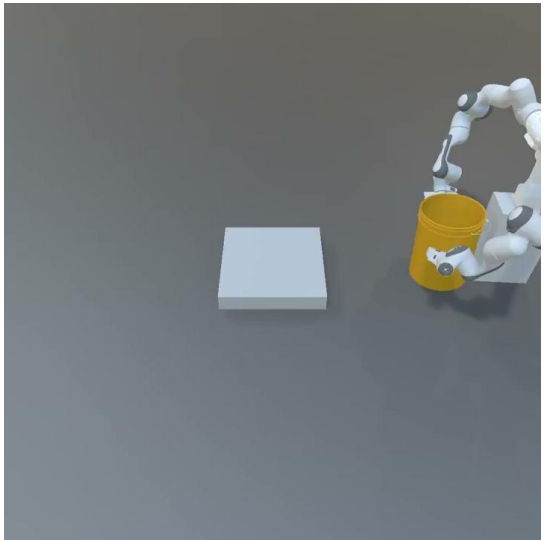
Goal specification: The target link (handle) is specified by the semantic mask

Success metric: The door/drawer is open (the joint state exceeds a threshold)

Task Specification



PushChair



MoveBucket

Goal specification: The goal position (red point or white platform) needs to be inferred from the RGB-D observation

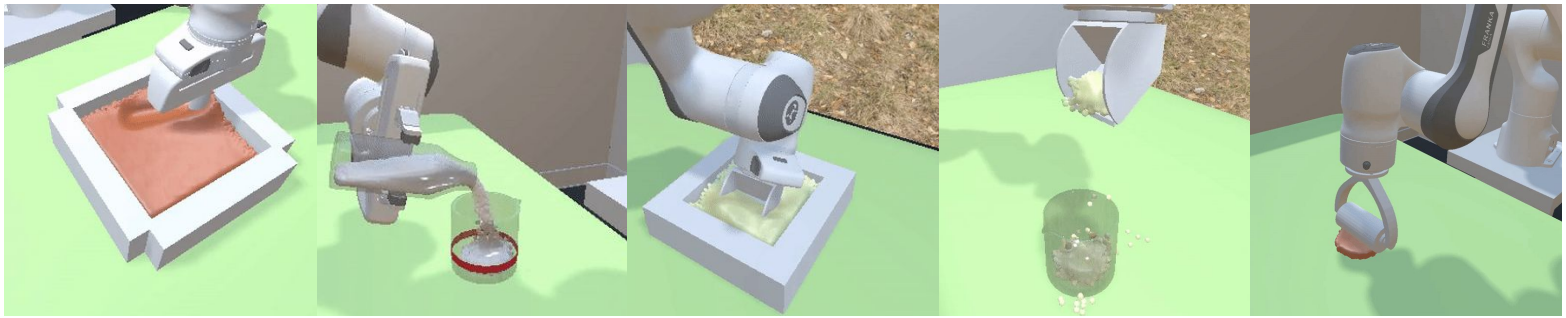
Success metric: The target object is located at the goal position

Assets: Objects

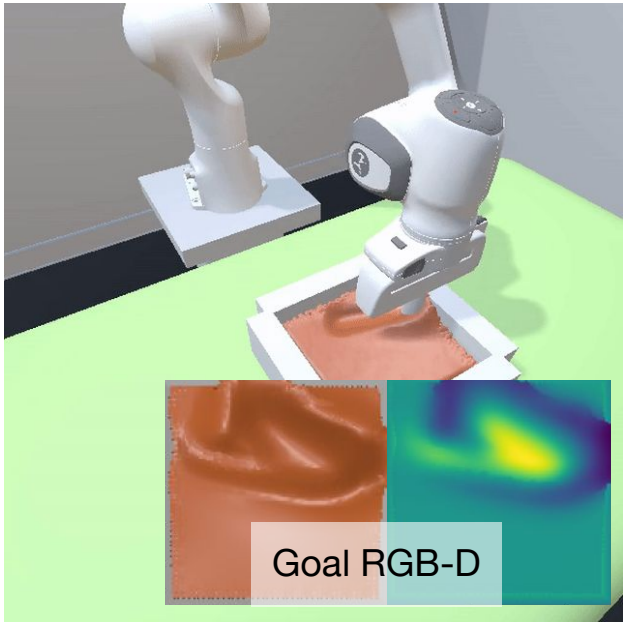
- 162 objects over 4 categories (from PartNet-Mobility)
- Large topology, geometry, and appearance variations

Case Study III*: ManiSkill-Softbody

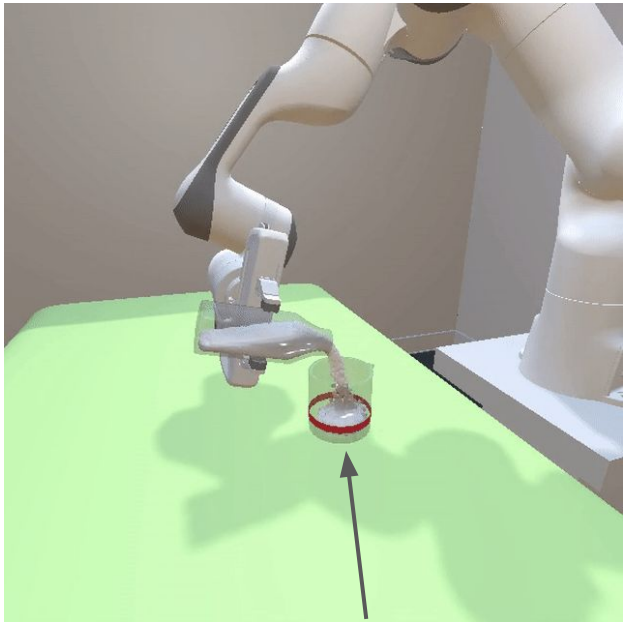
- Full physical rigid+soft body manipulation
- 5 deformable manipulation tasks (to be released in July)
- Embodiment: same as ManiSkill, except for special robot end-effectors — rod, bucket, and rolling pin



Goal Specification

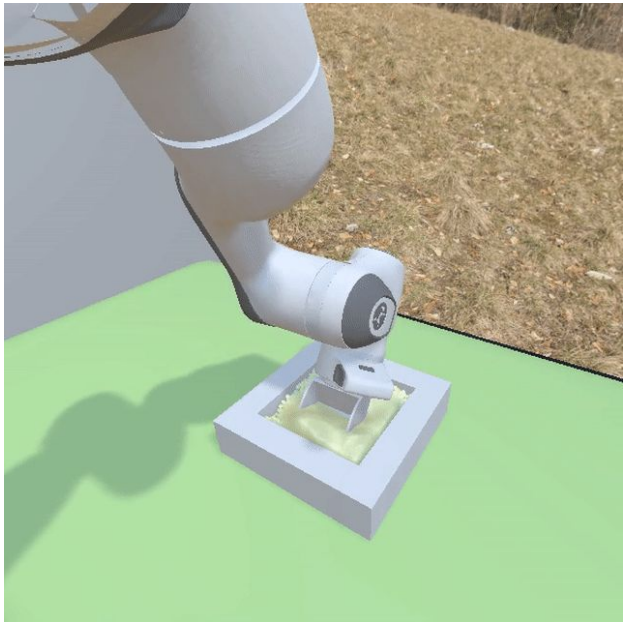


Writer: target shape specified as top-view RGB-D image

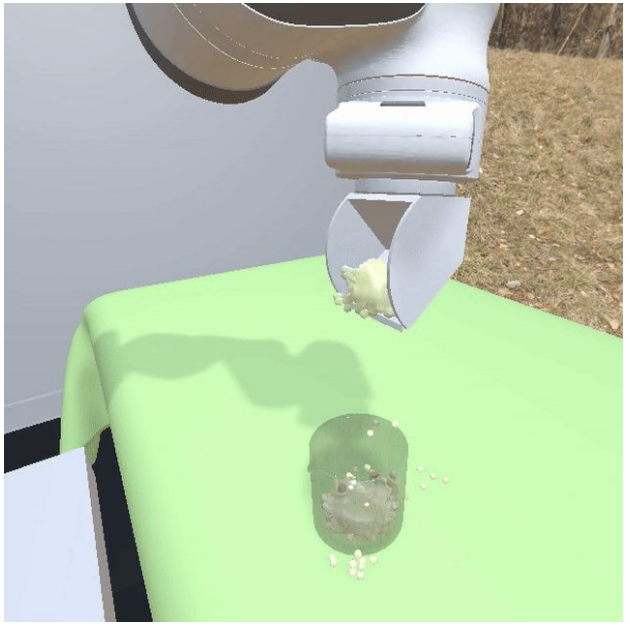


Pouring: desired fill level specified with the red marker ring

Goal Specification



Excavation: target scooped mass specified as a number



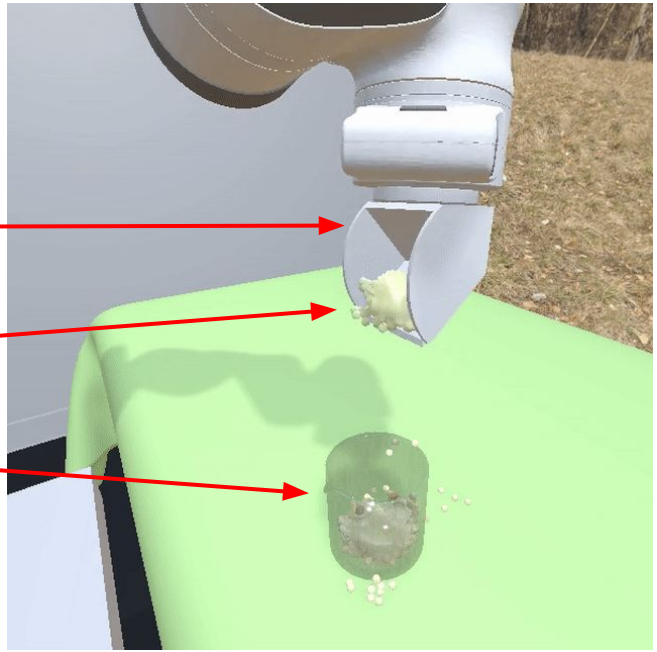
Filling: goal is inferred from placement of the beaker

Goal Specification

Rolling pin: target bounding box of the dough

Assets

- Special robot models
- Soft body represented by particles
- Rigid body containers



Summary

Name	Task Focus	Navigation actuation	Manipulation actuation	Assets
Habitat 1.0	Navigation	Non-physical	N/A	Static scenes
Habitat 2.0	Mobile manipulation	Non-physical	Partially physical	Interactive scenes
iTHOR	Language grounding Interactive navigation	Non-physical	Non-physical	Interactive scenes
ManipulaTHOR	Mobile manipulation	Non-physical	Partially physical	Interactive scenes
ManiSkill	Object manipulation	Physical	Physical	Articulated objects

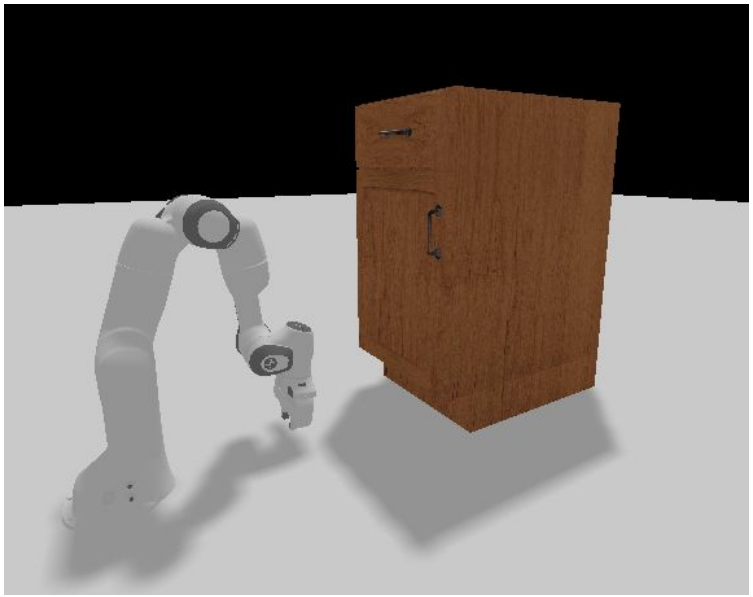
Summary

- Low-level vs. high-level
- Physical vs. non-physical
- Scene-level vs. object-level

ManiSkill 2022 (incoming July)

Embodiment

- Sensor subsystem
- Actuator subsystem



Sensor Subsystem

- Visual perception signals
- Robot proprioception signals
- Haptics and other signals

Embodiment: Sensor

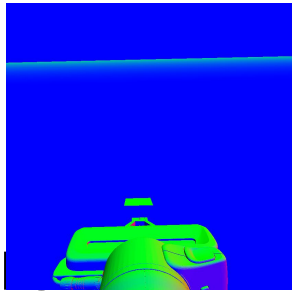
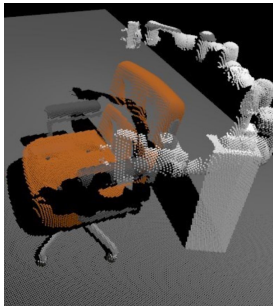
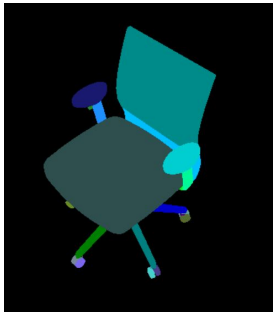
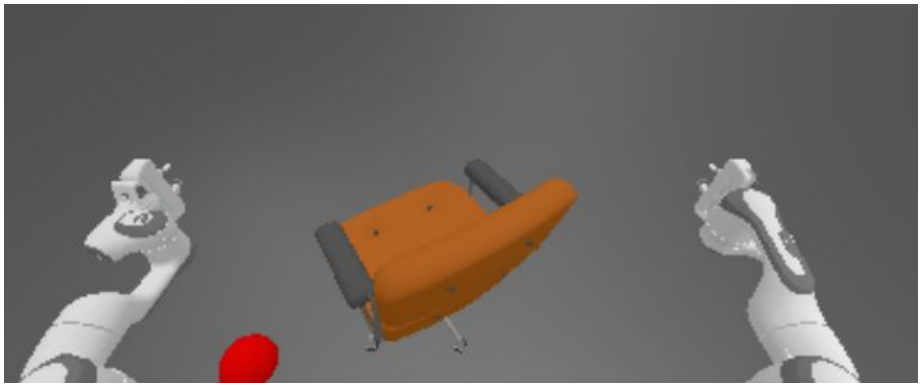
- Common types of sensor signals
  - Visual perception
  - Robot proprioception
  - Haptic and tactile perception

RGB and depth sensors

Intel RealSense D455

Visual Perception Signals

- Types
  - RGB
  - Depth & normals
  - Semantic/instance mas
  - Optical/scene flow
- Sensor placement
  - Head
  - Arm
  - Third-view

Haptics and Other Signals

- Force sensors
- Grasp feedbacks (magic or real grasp)
- GPS and IMU (compass)
- …

Actuator Subsystem

- Mechanical components
  - Mobile platform, arm, gripper
- Controllers
  - How to control each robot component
  - Action space (continuous vs. discrete)

Mechanical Components of Actuators

- Mobile platform
- Arms
- Grippers

Embodiment: Actuator

- Common types of input to actuator
  - High-level command
    - E.g., “move the end-effector forward”
    - Usually not physically-based, partial degree of freedom
  - Intermediate-level motor command
    - E.g., end-effector position and orientation
    - Physically-based, partial degree of freedom
  - Low-level motor command
    - E.g., torque of each joint
    - Physically-based, full degree of freedom

Initial States

- Robot states
  - Initial base pose
- Joint states
  - Initial poses
- Objects
  - Properties (appearance & internal states)
  - Poses
- Environment properties
  - Textures, lighting, etc. (for domain randomization)

Metrics

- Scalar vs. Binary
- Examples
  - Rearrangement: check pose
  - Return to home position
  - Velocity requirement: robot or object
  - State change: e.g. is cooked

Physical vs. Non-physical Simulation

- Sensors
- Controllers
- Interactions with objects

Design Choices in Modern Embodied AI Environments

- Embodiment
  - Sensors
  - Actuators
- Task Specification
  - Goals
  - Metrics
- Objects & Scenes
  - Scale and diversity
  - Supported states & properties
- Physical vs. non-physical simulation

How to Implement Mobile Platform?

- Dynamic
  - Augment existing URDF to add dummy base joints
  - Controlled by joint velocity
- Kinematic
  - Directly set the robot pose
  - Controlled by delta joint position
  - Habitat uses navmeshes to take collision between into consideration

Arm Controllers

- End-effector space
  - Delta pose
- Joint space
  - Delta position
  - joint velocity

Grasp Controllers

- Magic grasp
  - Binary
  - Mask-based
- Real grasp
  - Joint position
  - Mimic
- Vacuum grasp

Embodiment: Actuator

- The type of input command defines the action space
  - High-level commands usually correspond to discrete actions
  - Low-level commands correspond to continuous actions
- Different robot components like arm and gripper can be equipped with different actuators

Case Study II-B: AI Habitat 2.0

- Habitat 2.0
  - Home Assistant Benchmark (HAB): focus on mobile Manipulation

The figures and videos in this section are from Habitat 2.0 unless specified

Embodiment: Robot

- Fetch: mobile platform (wheeled) + 7 DoF arm
- The height of torso link is adjustable, but fixed in the HAB

Figure from Fetch

Embodiment: Visual Sensors

- Camera types
  - RGB
  - Perfect depth
    - Depth is normalized and clipped to [0m, 10m]
  - Perfect semantic instance masks
- Camera positions
  - Mounted on the head or arm (wrist)
  - Third-view (debug/visualization only)

Embodiment: Visual Sensors

Third-view RGB

Head RGB

Head depth

Arm RGB

Arm depth

Embodiment: Proprioception

- Perfect GPS and compass
- Joint positions and velocities
- End-effector position (3D) in the base frame
- Whether an object is grasped in the end-effector

Controllers: Base

- Continuous velocity control
  - Linear and angular velocity
  - Implementation: kinematically setting the robot base on navmesh for collision
- Discrete actions can also be defined and converted to continuous actions

Cons of Kinematic Base Control

Cons: arm collision is not taken into consideration during base navigation

Controllers: Arm

- Delta end-effector position
  - The action is a 3-dim vector to indicate the desired displacement of **the end-effector in the Euclidean space**
  - Implemented by computing desired joint positions with position-only Inverse Kinematics (IK). The target end-effector position is clipped to a predefined workspace.
- Faster to learn and easier for human to manually control

Controllers: Arm

- Delta joint position
  - The action is a 7-dim vector to indicate the desired displacements of **joint positions (angles) in the configuration space**
- More flexible and can tackle harder tasks, like collision avoidance

Controllers: End-effector

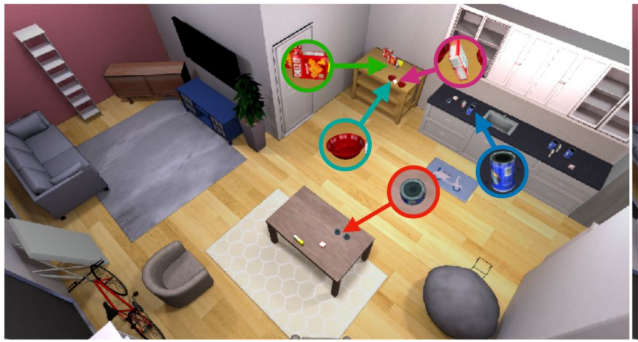
- Magnet-based magic grasp
  - Scalar action: positive to grasp and negative to release
  - The closet object within 15cm from the gripper will be grasped if a positive signal is given
  - The grasped object will be snapped to the end-effector by a predefined relative position (for all objects)
- It is not suitable for precise placement as users can not fully control the behavior

Cons of Magic Grasp

Cons: orientation can not be controlled

Task Specification

- GeometricGoal: starting and desired 3D positions of target objects. Orientation is not considered.
- An episode is successful *if all target objects are placed within 15cm of the goal position for the object*.



Starting positions of 5 objects

Desired positions of 5 objects

Assets: Scenes

- ReplicaCAD
  - 105 scenes redesigned by artists from Replica dataset
  - 5 macro variations (different layouts of large furniture) with 21 micro variations (w.r.t. small furniture) per macro variation

5 macro variations

Assets: Objects

- Rigid objects: YCB dataset kitchen and food categories
- Articulated objects: fridge and kitchen counters
- The initial poses of objects are generated offline
  - Require annotated surfaces of receptacles to place objects
  - Objects are randomly placed on receptacle surfaces without collision with rejection
  - Reject unstable configurations

Initial States

- Robot states
  - Initial position and orientation are randomly and collision-free per episode
  - A predefined resting pose for arm. Noise can be added
- Objects
  - Initial pose of objects and targets are generated offline to verify feasibility of the episode
  - Joint states of the articulated are task-dependent, but usually resting state (defined in the URDF)

Embodiment: Actuator

- Kinematic base movement only, no physical interaction
- 4 discrete actions: move forward (0.25m), turn left or right (10 degree) and stop
- **Navmesh** is generated per static scene (usually offline), which can be used to compute navigable positions at a given height efficiently.

Case Study III: ManiSkill

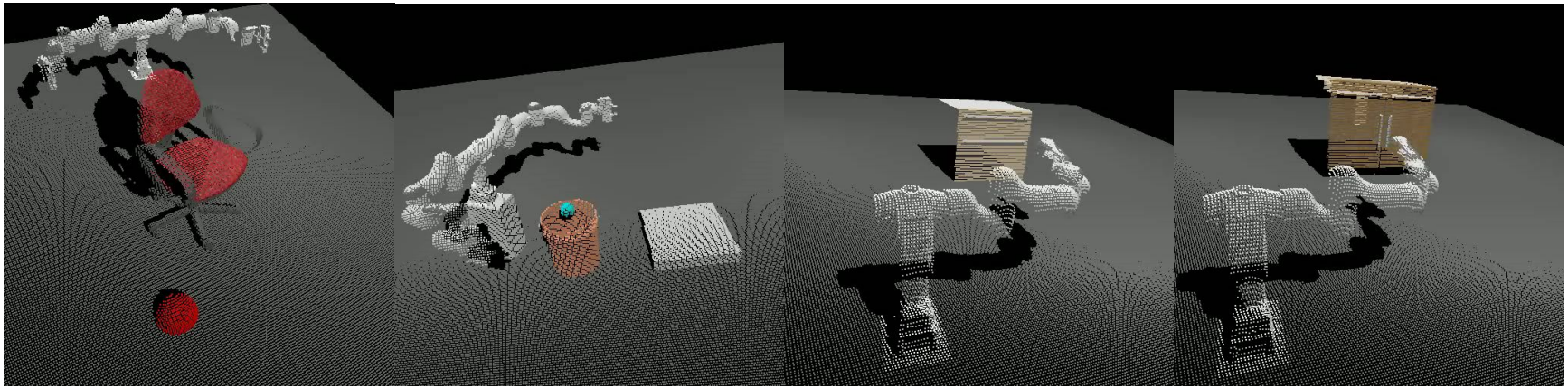
- ManiSkill
  - Physical Manipulation
  - Object-level Generalization

Embodiment: Visual Sensors

- Camera types
  - RGB
  - Perfect depth
  - Perfect masks of robot and some object parts
    - E.g., The masks of target door and handle are provided in OpenCabinetDoor task
- Camera positions
  - Mounted on the top of the robot, moving with the robot
  - The three cameras are 120° apart from each other

Task Specification

- OpenCabinetDoor/Drawer: Target specified by mask
- PushChair: Target position specified by a red dot
- MoveBucket: Target is a white platform



Initial States

- Robot states
  - Initial position, rotation & poses
  - Egocentric visual observation (not directly specified)
- Objects
  - Initial pose of the target object
  - Joint states and internal states (not directly specified)
- Environment properties
  - Textures, lighting, etc. (observed but not directly specified)

Embodiment: Actuator

- Examples of different input and output commands

Input command	Example	Degree of freedom under control	Output command
High-level	“move the base forward 0.1m”	Partial	Usually kinematic
Intermediate	Target end-effector pose	Partial	Dynamic
Low-level	Torque on each joint	Full	Dynamic

Actuator: Example

Base: set the pose of the robot

Arm: motor control given target end-effector movement

Gripper: a magnet doing abstract grasp

Embodiment: Actuator

- Examples of different input and output commands

Input command	Example	Degree of freedom under control	Output command
High-level	“move the base forward 0.1m”	Partial	Usually kinematic
Intermediate	Target end-effector pose	Partial	Dynamic
Low-level	Torque on each joint	Full	Dynamic

Case Study II: AI2THOR

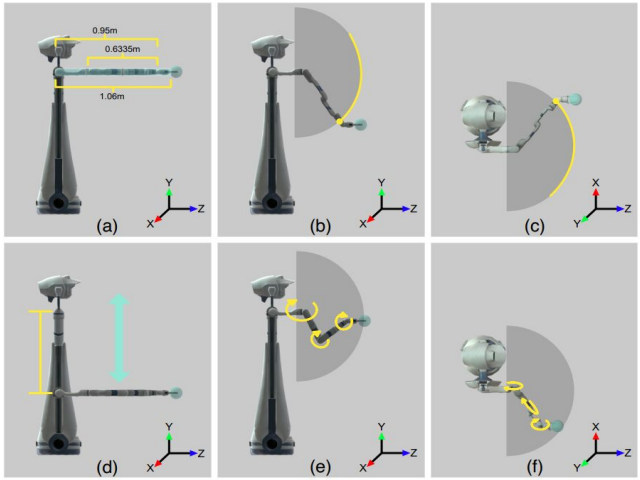
- iTHOR:
  - Navigation and high-level object interaction
  - Instruction following
  - Embodied VQA
- ManipulaTHOR:
  - Mobile manipulation (pick-and-place)

iTHOR

ManipulaTHOR

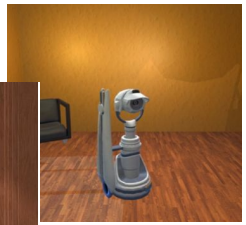
Embodiment: Robot

- Arm with adjustable height and length
- 6 DoF wrist with magic grasp



Embodiment: Visual Perception

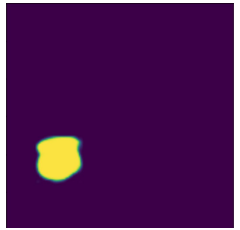
- Types of sensor s
  - RGB-D
  - Instance masks (
  - 3D bounding box ning)
- Camera placemer
  - M
  - Fl ne



tion are from AI2THOR unless specified

Embodiment: Actuator

- Base
  - Discrete delta position & rotation
- Arm
  - Target pose (for ManipulaTHOR)
  - Discrete delta pose (for ManipulaTHOR)
- Gripper
  - Mask-based primitive actions (for iTHOR)
  - ne magic grasp (for ManipulaTHOR)



Task: Goal Specification

- Geometric:
  - Coordinates of target (ArmPointNav)
- Semantic
  - Category of the target (RoboTHOR Challenge)
- Language instructions
  - High-level or step-by-step human instructions (ALFRED Challenge)
- Visual observations
  - The agent has to reason visually (Rearrangement Challenge)