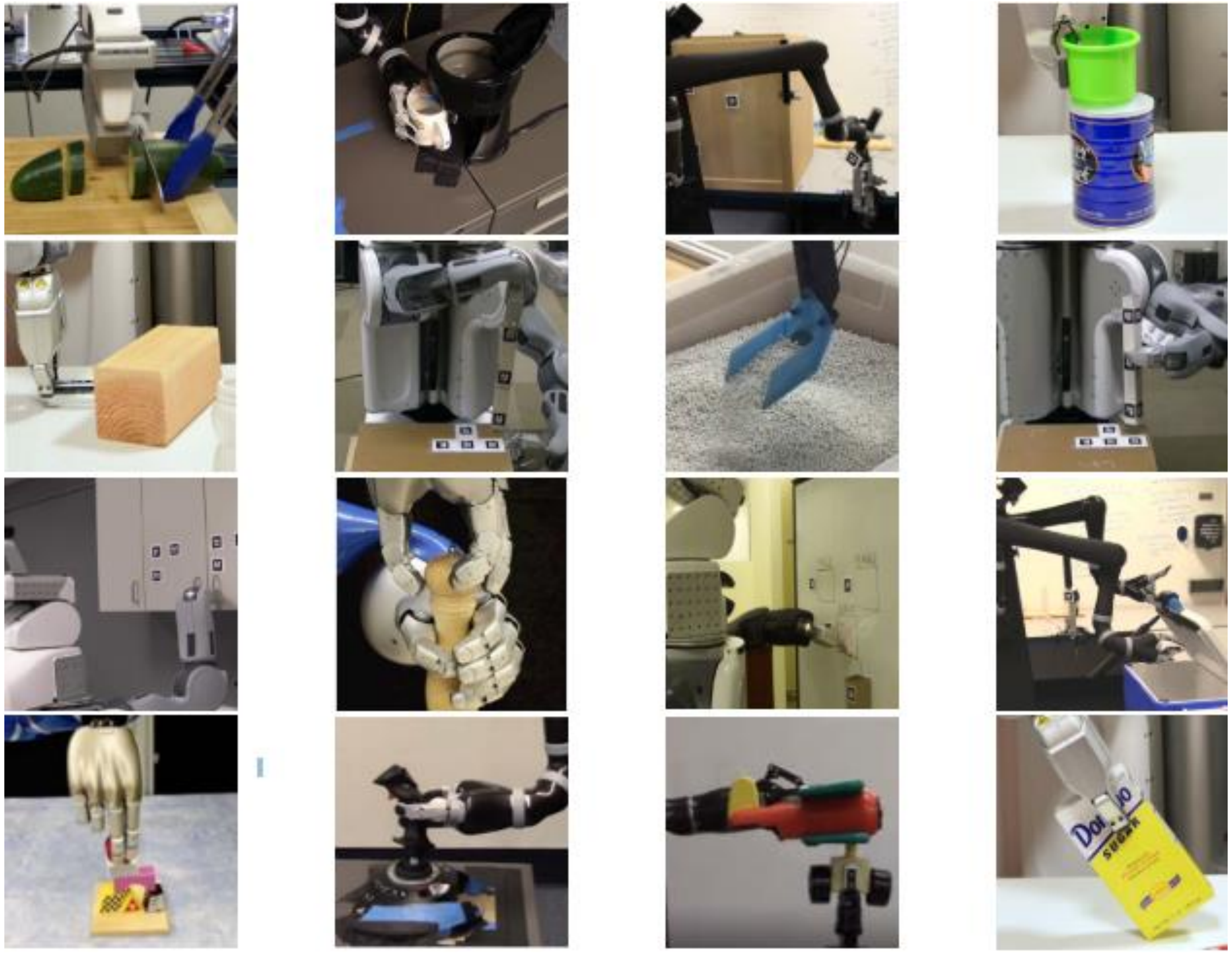
AI for learning tasks / movements

Object Representations

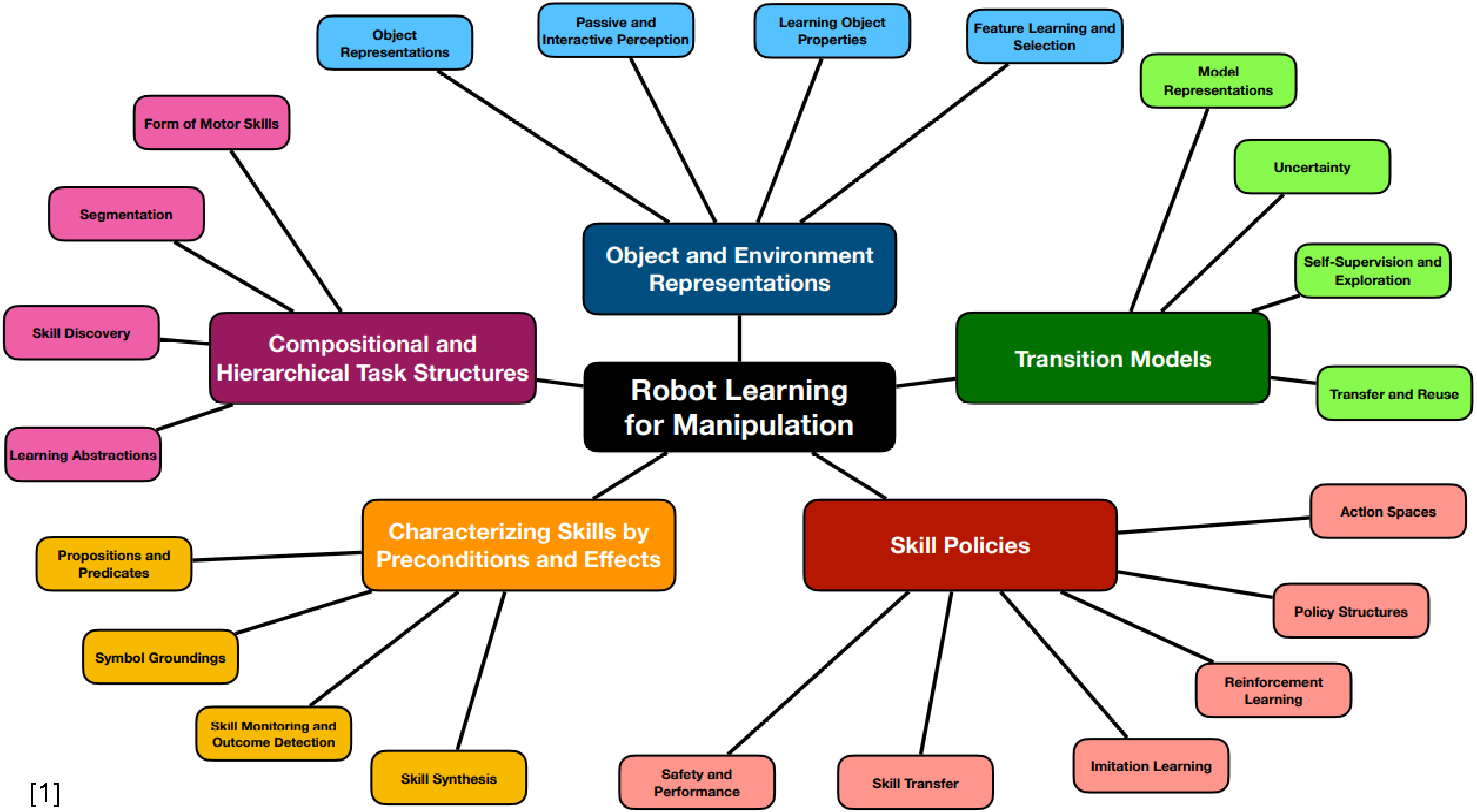
Transition Models

Skill Policies

Hierarchical Task Structures



[1] Example manipulation skills including inserting, stacking, opening, pushing, cutting, screwing, pouring, and writing.



Object Representations
Passive and Interactive Perception
Learning Object Properties
Feature Learning and Selection
Model Representations
Form of Motor Skills
Uncertainty
Segmentation
Object and Environment Representations
Self-Supervision and Exploration
Skill Discovery
Compositional and Hierarchical Task Structures
Transition Models
Robot Learning for Manipulation
Transfer and Reuse
Learning Abstractions
Action Spaces
Characterizing Skills by Preconditions and Effects
Skill Policies
Propositions and Predicates
Policy Structures
Symbol Groundings
Reinforcement Learning
Skill Monitoring and Outcome Detection
Imitation Learning
Skill Synthesis
Safety and Performance
Skill Transfer

[1]

Object and Environment Representations

Object Representations

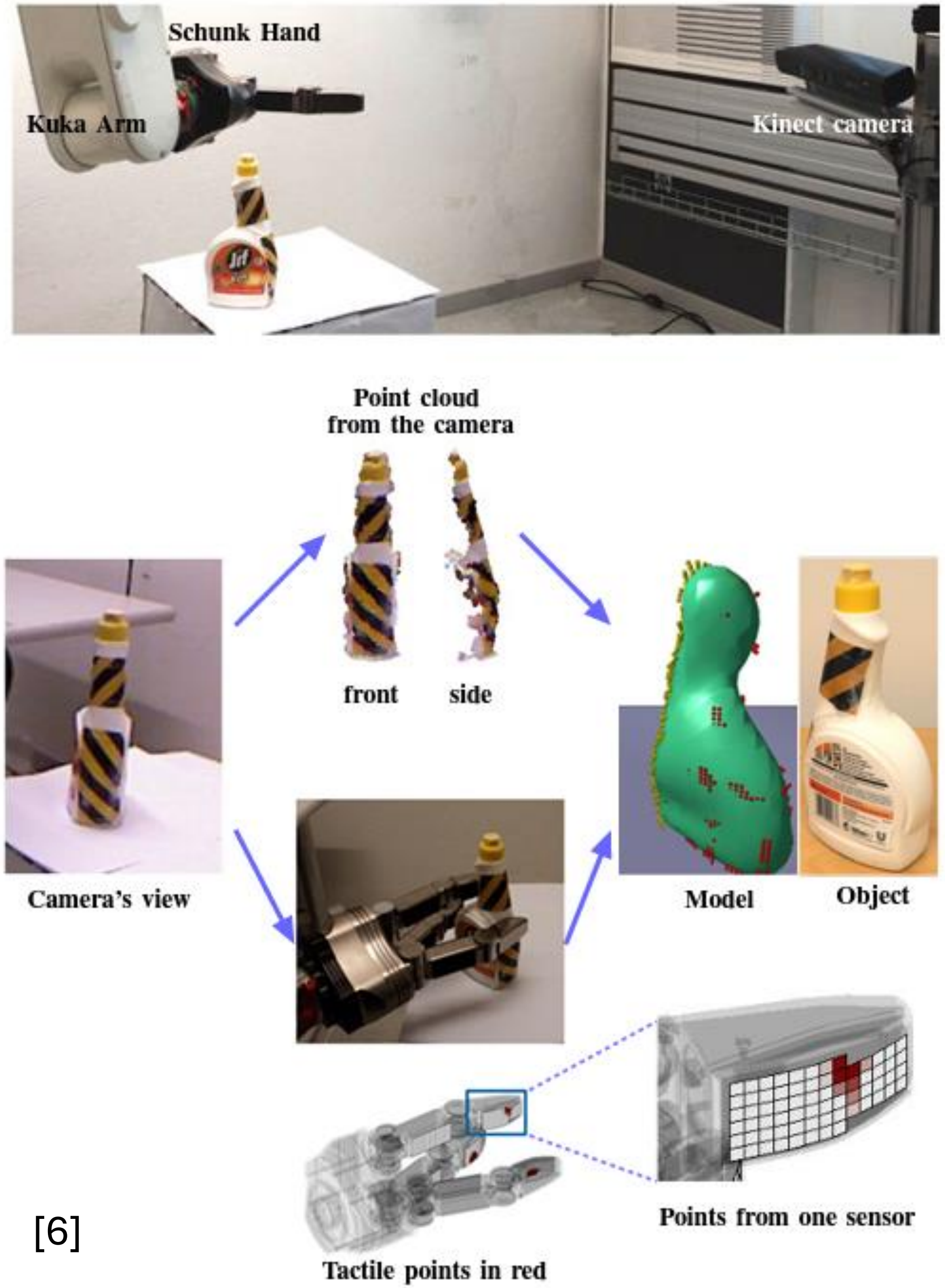
Passive
and Interactive Perception

Learning Object Properties

Feature Learning and Selection

Object Representations

- Point Level
- Part Level
- Object Level



[6]

Learning Object Properties

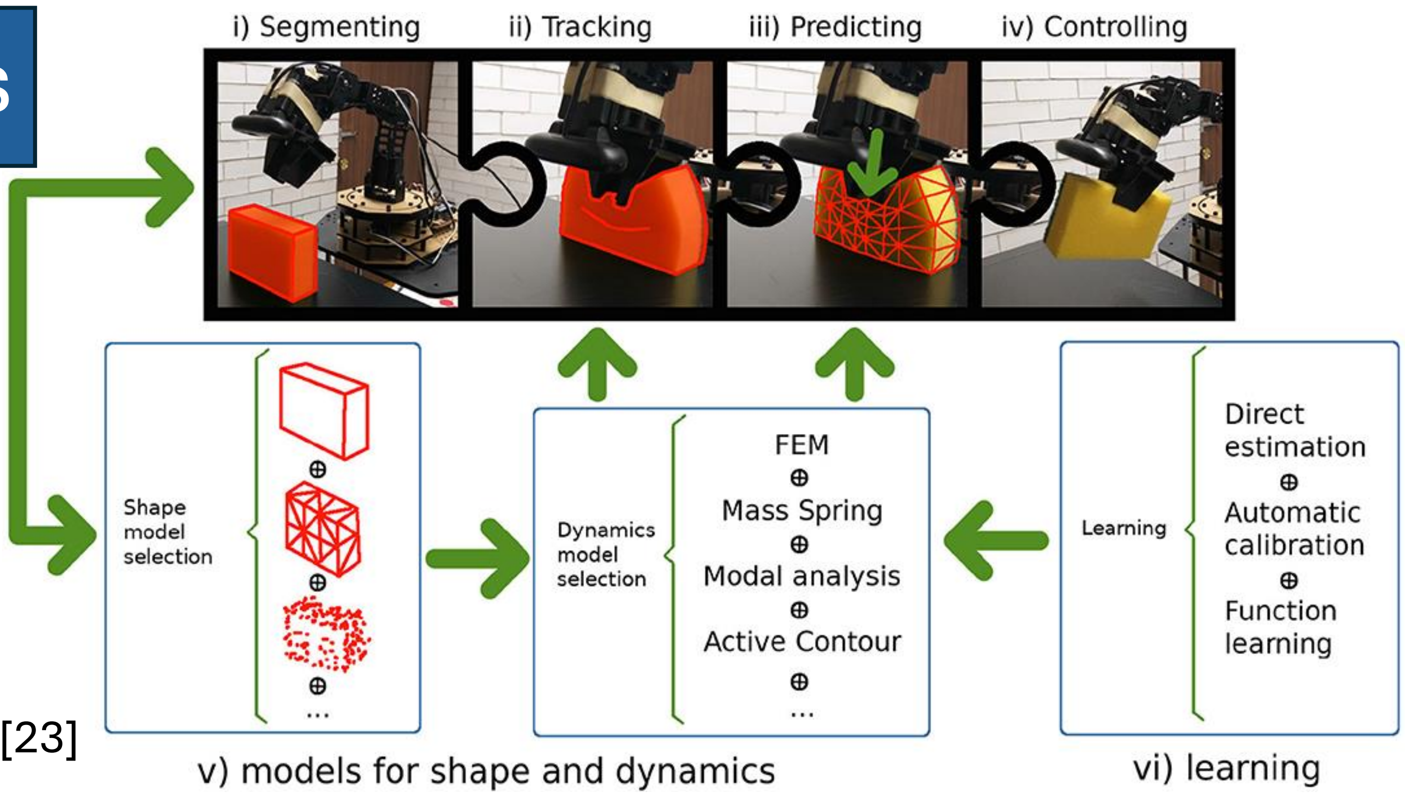
Shape

Pose

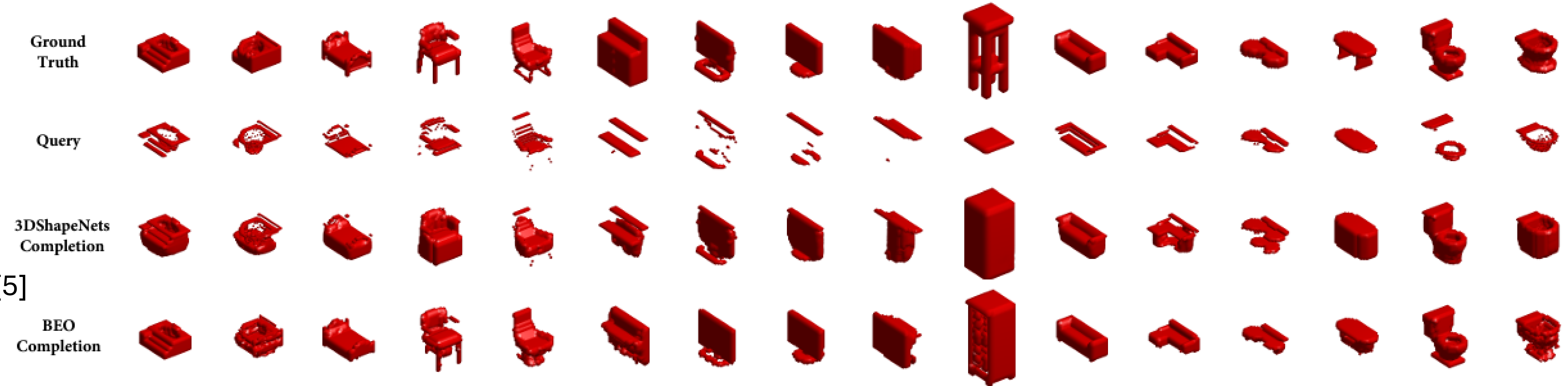
Material

Physical Constraints

Dynamic Properties



[23]



[5]

Passive and Interactive Perception

Passive:

- Using camera to observe the environment
- Moving a camera to a better vantage point
- Observe human performing an action

➕ Lots of observation in short time

➕ Low effort

Interactive:

- Push a box to obtain its properties
- Haptic, tactile, vision, and audio sensors

➕ More specific details

➕ Supervisory signal for passive perception

Transition Models

Model Representations

Uncertainty

Self-Supervision and Exploration

Transfer and Reuse

Model Representations

Deterministic Function:

$$T : S \times A \rightarrow S$$

Stochastic Distribution:

$$T : S \times A \times S \rightarrow \mathcal{R}$$

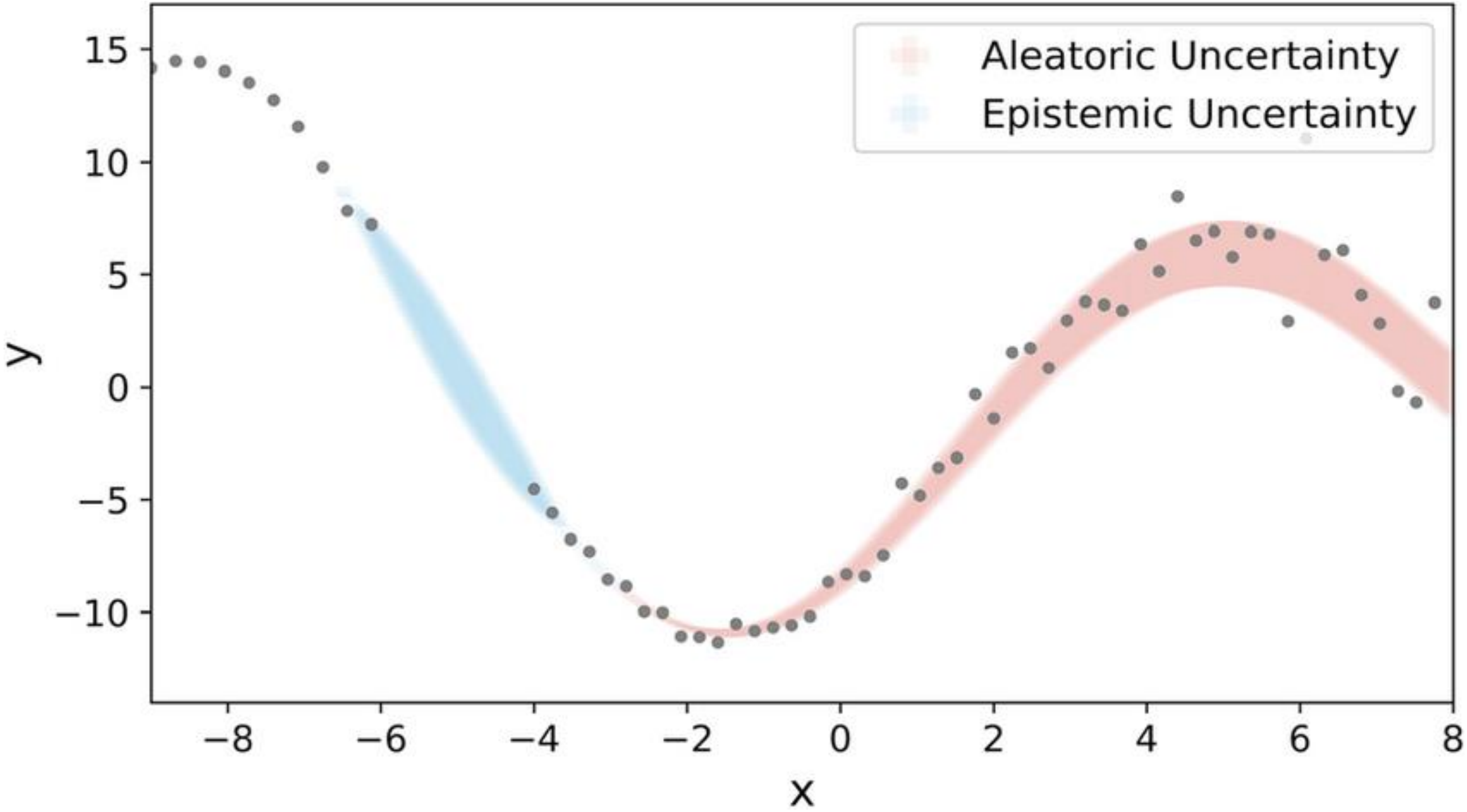
[1]

Continuous Models

Discrete Models

Hybrid Models

Uncertainty



[20]

Self-Supervision and Exploration

Random Sampling

Active Sampling

Grid Sampling

Intrinsic Motivation

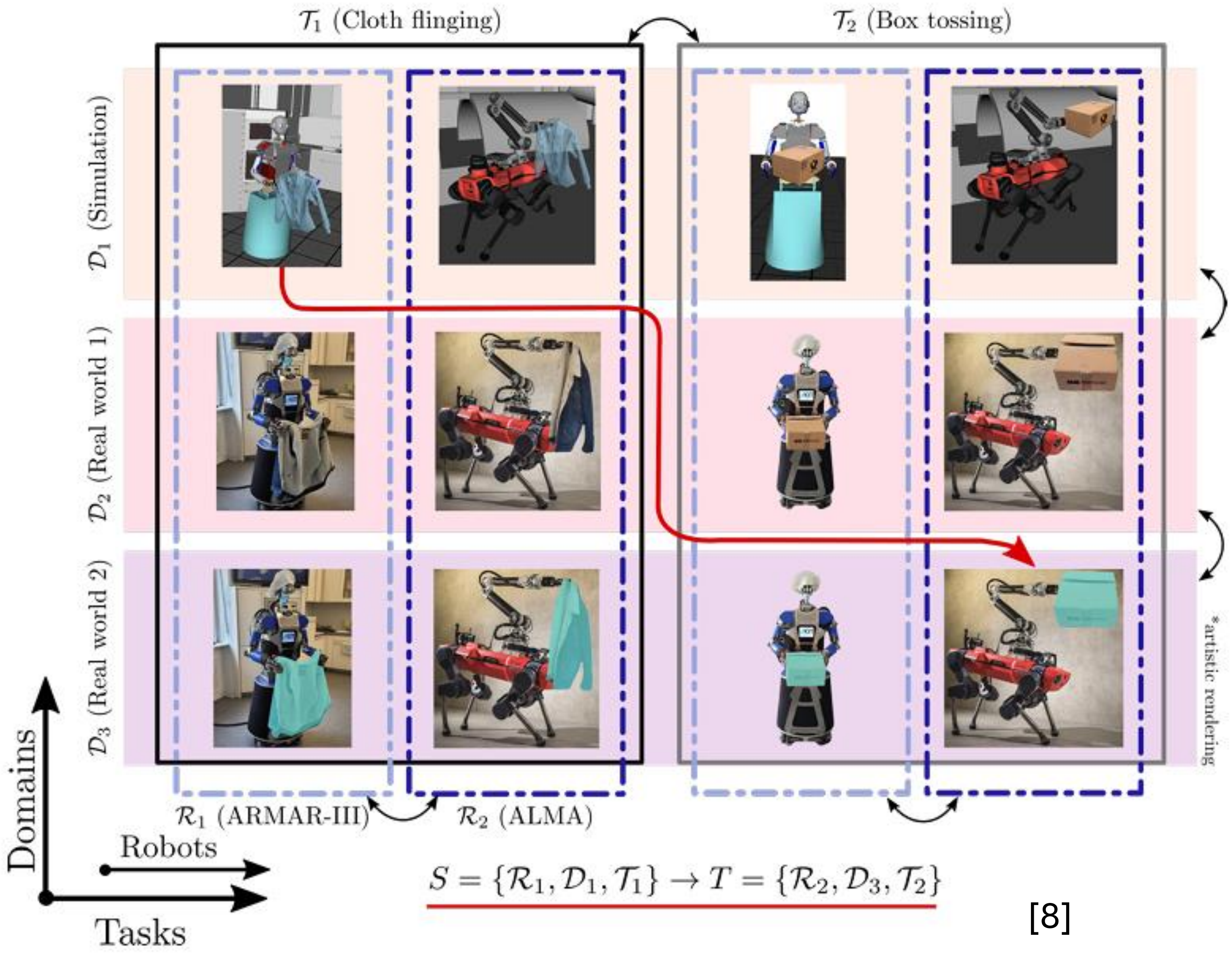
Transfer and Reuse

Conditions for Model Reuse:

- **Compatibility Required**: Same state, action, and context spaces.
- **Mapping Necessity**: If spaces differ, a mapping is required to align them.

Challenges in Model Transfer:

- **Covariate Shift**: Variability in input data distribution between tasks.
- **Dataset Shift**: Variability in both input and output data distributions.



[8]

Skill Policies

- Action Spaces
- Policy Structures
- Reinforcement Learning
- Imitation Learning
- Skill Transfer
- Safety and Performance

What are Skill Policies?

- **Goal:** aquire *skill controller*
- Common *skill controller* representation:

    State-action pairs mapped to probabilities

Let's see how to implement skill policies!

Skill Policies:

Action Spaces

Action spaces

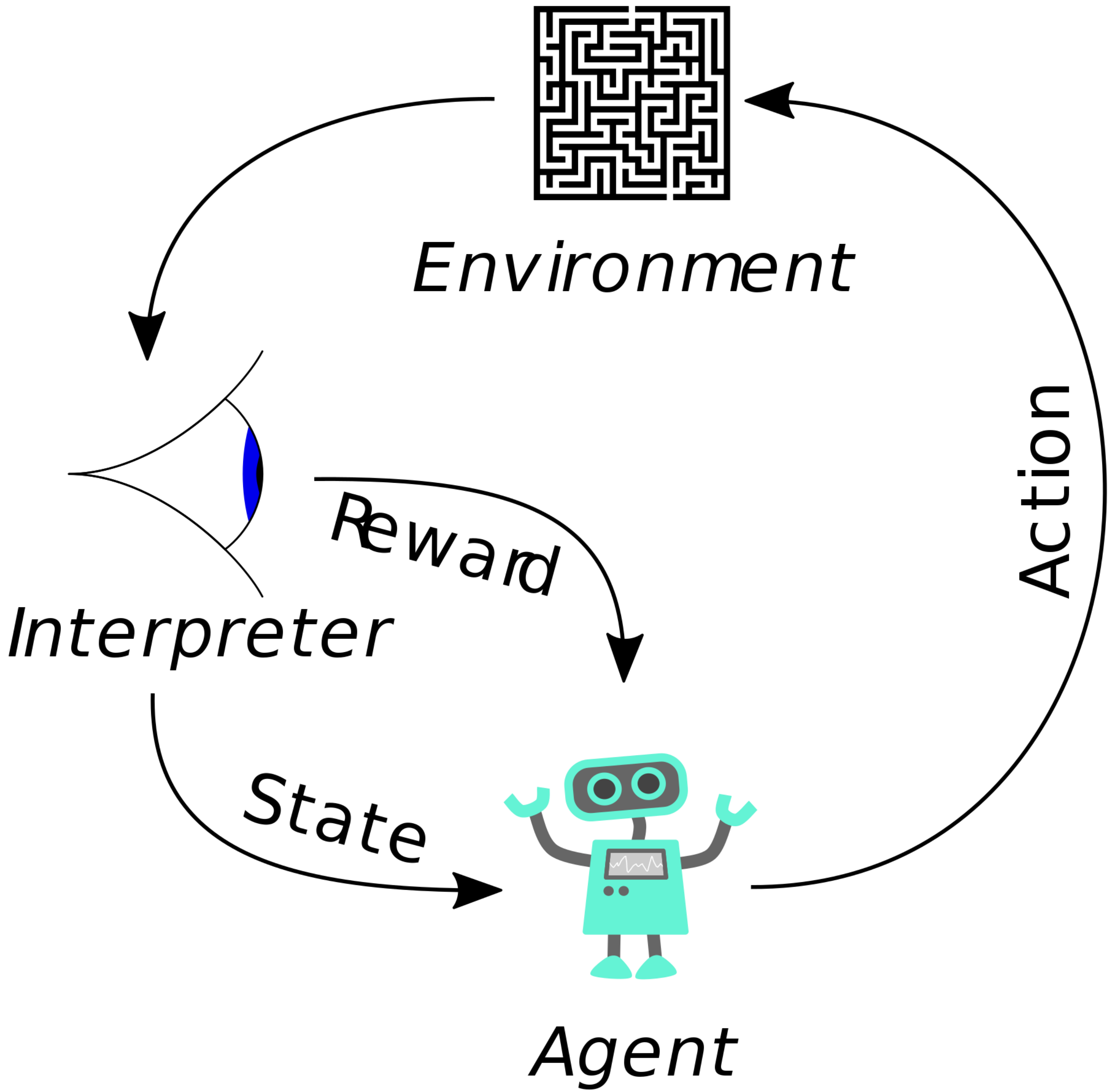
- When implementing Skill Policy, you need to select an Action Space (a set of valid robot actions)
- An intermediary controller is often placed between the policy output and actuators (actuators can be, for example, motors)
- Intermediary controller translates the Action into specific signals for the actuators
- This is necessary for complexity reduction and stability.

Skill Policies:

Reinforcement learning

What is Reinforcement Learning?

Subfield of ML where an agent (robot) learns to make decisions by performing actions in an environment to maximize cumulative reward.



[2]

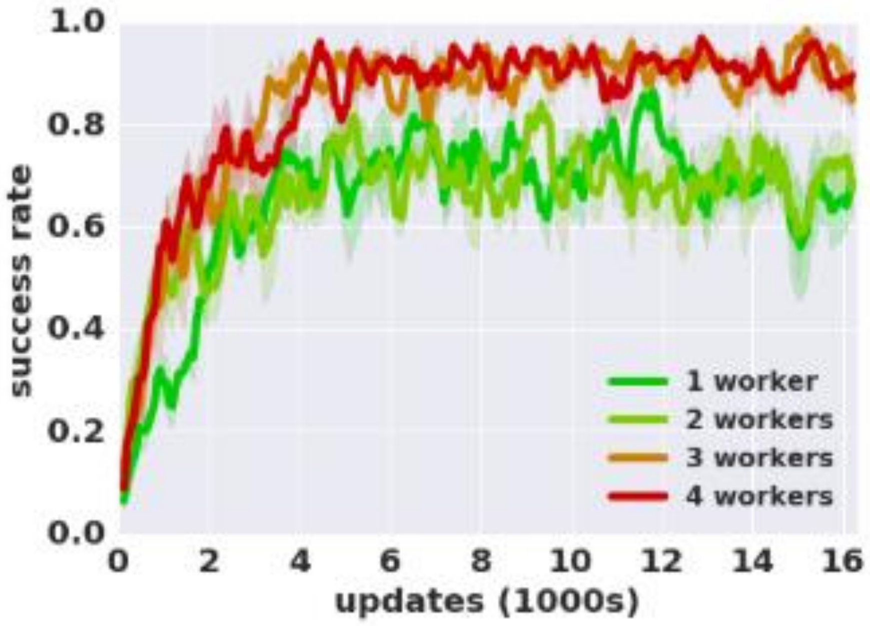
Student-teacher training

- **Teacher Policy:**
- Has access to all domain parameters
- Adapt behaviors based on the specific dynamics of each simulated scenario

- **Student Policy:**
- No direct access to domain parameters
- Learns by mimicking the teacher's actions
- Uses historical sensory data to infer the necessary parameters for decision-making

Asynchronous Training

- The speed of learning in asynchronous training depends on the ratio of data collection speed (by the collectors) to the training speed (by the trainer).
- Multiple robots performing tasks can send back a variety of situational data to a central model for faster learning and adaptation.



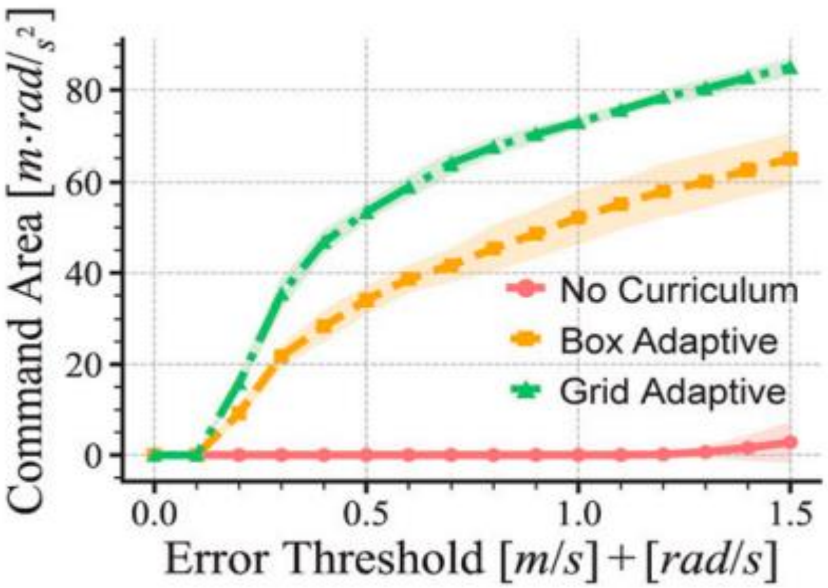
(a) Reaching

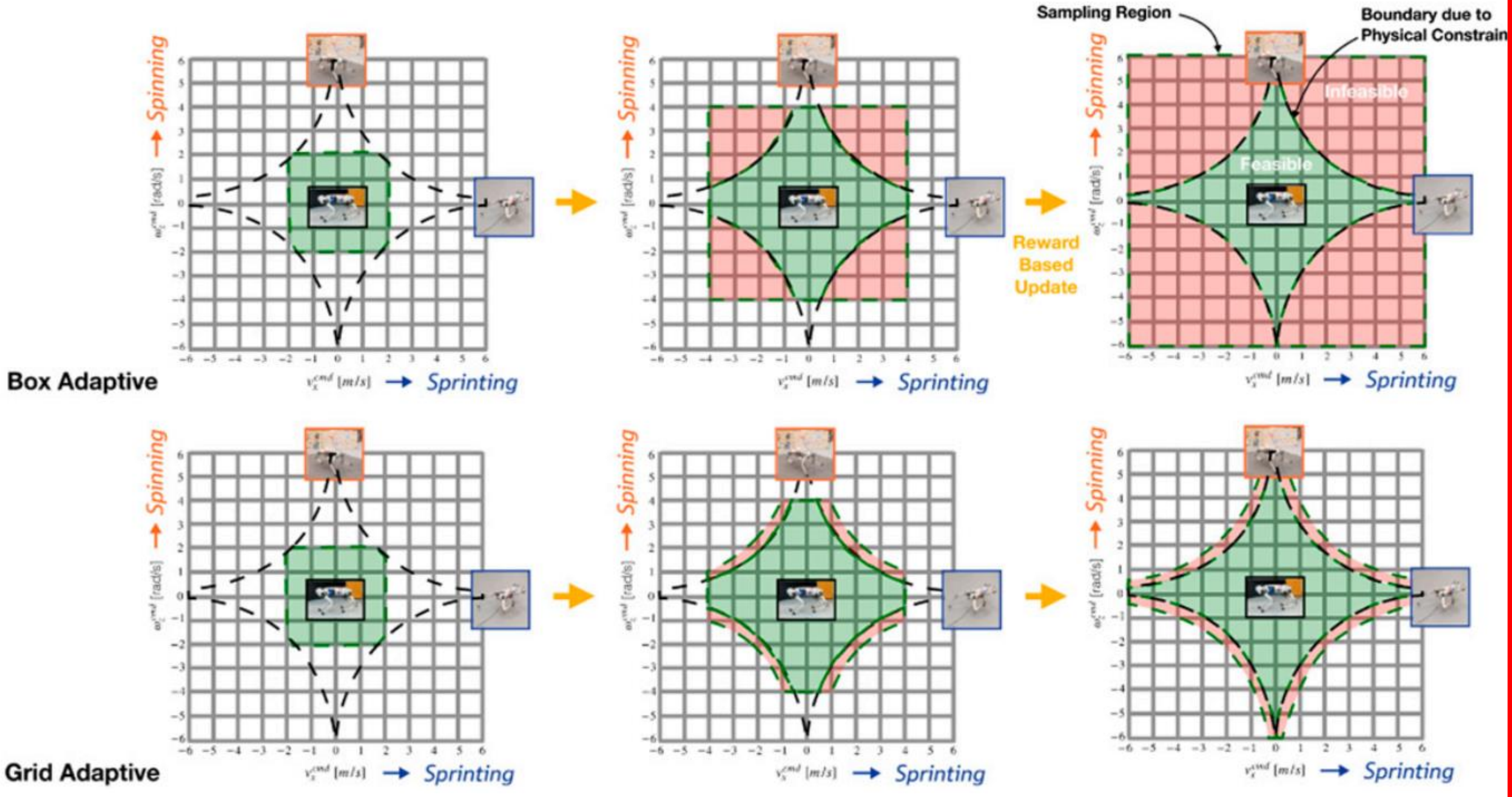
[3] (b) Door Pushing

Curriculum Learning

Train the model with a sequence of examples of increasing difficulty

Automated curricula generation using GANs





[2]

Curiosity-Driven Learning

Mechanisms to Guide Exploration:

Expert Demonstrations	Requires predominantly hand-crafted demonstrations
Curriculum Learning	Generation and efficient scheduling of intermediate tasks are still considered unsolved
Intrinsic Motivation (Curiosity)	Learn without external rewards for the pure sake of knowledge gain.

How to incorporate curiosity?

1. Surprise-Based Curiosity

- **Approach**: Model that predicts the environment's forward dynamics. The intrinsic reward is then based on the Euclidean distance between the predicted next state and the observed transition.

2. State Embedding Predictions

- **Approach**: Rather than predicting the complete world state directly, predict a lower-dimensional embedding of the state.

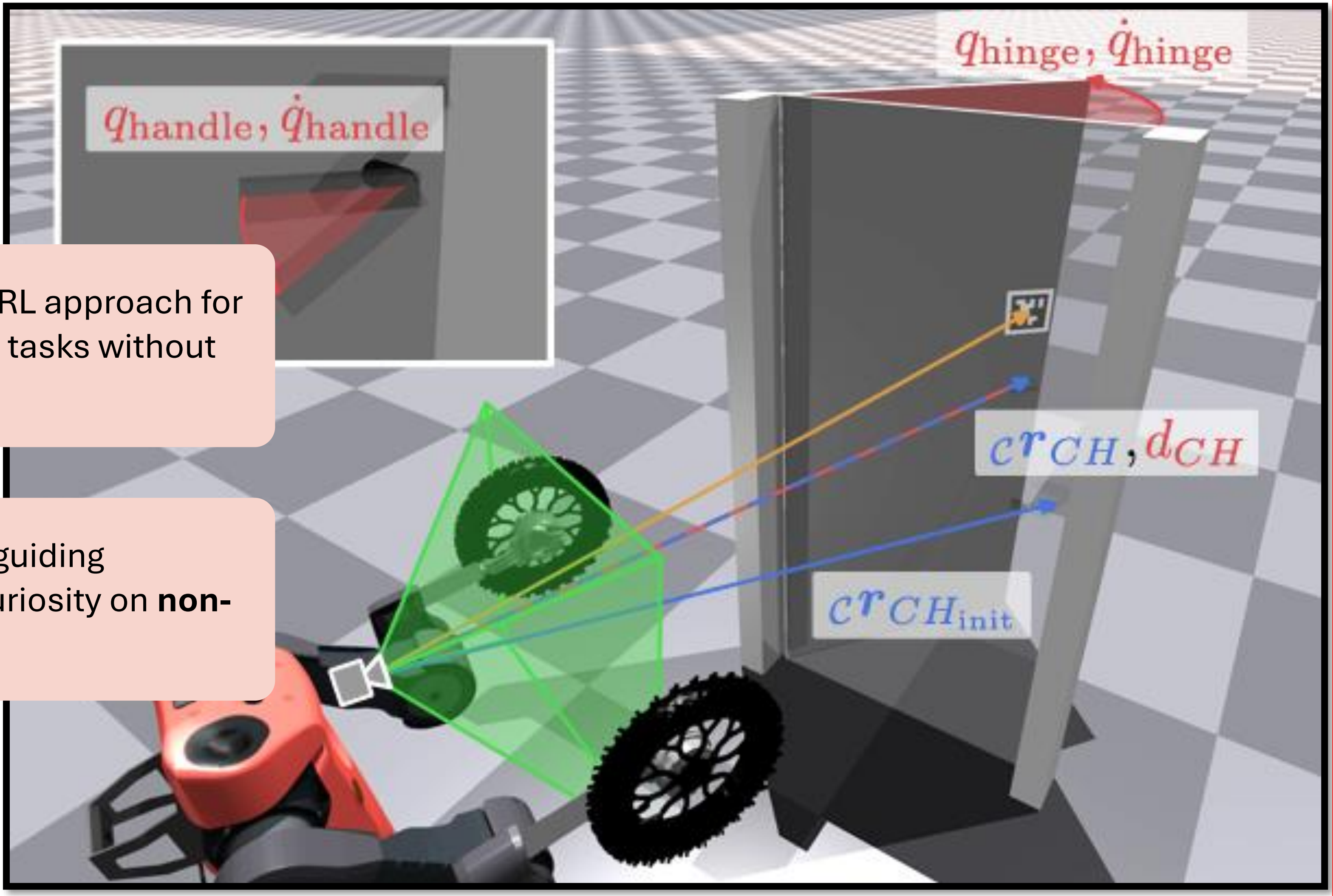
3. Learning Progress and Count-Based Methods

- **Learning Progress**: Intrinsic rewards are given for actions that lead to improvements in the agent's understanding or capability within particular areas of the state space.
- **Count-Based Exploration**: Implements a straightforward count of visits to each state, with a preference for states that have been visited less frequently.

[7], [9]

A curiosity-driven **sparse reward** RL approach for learning end-to-end manipulation tasks without **task-specific engineering**

The notion of **curiosity states** as guiding mechanisms, allowing to focus curiosity on **non-directly observable states**



[9]


RSL

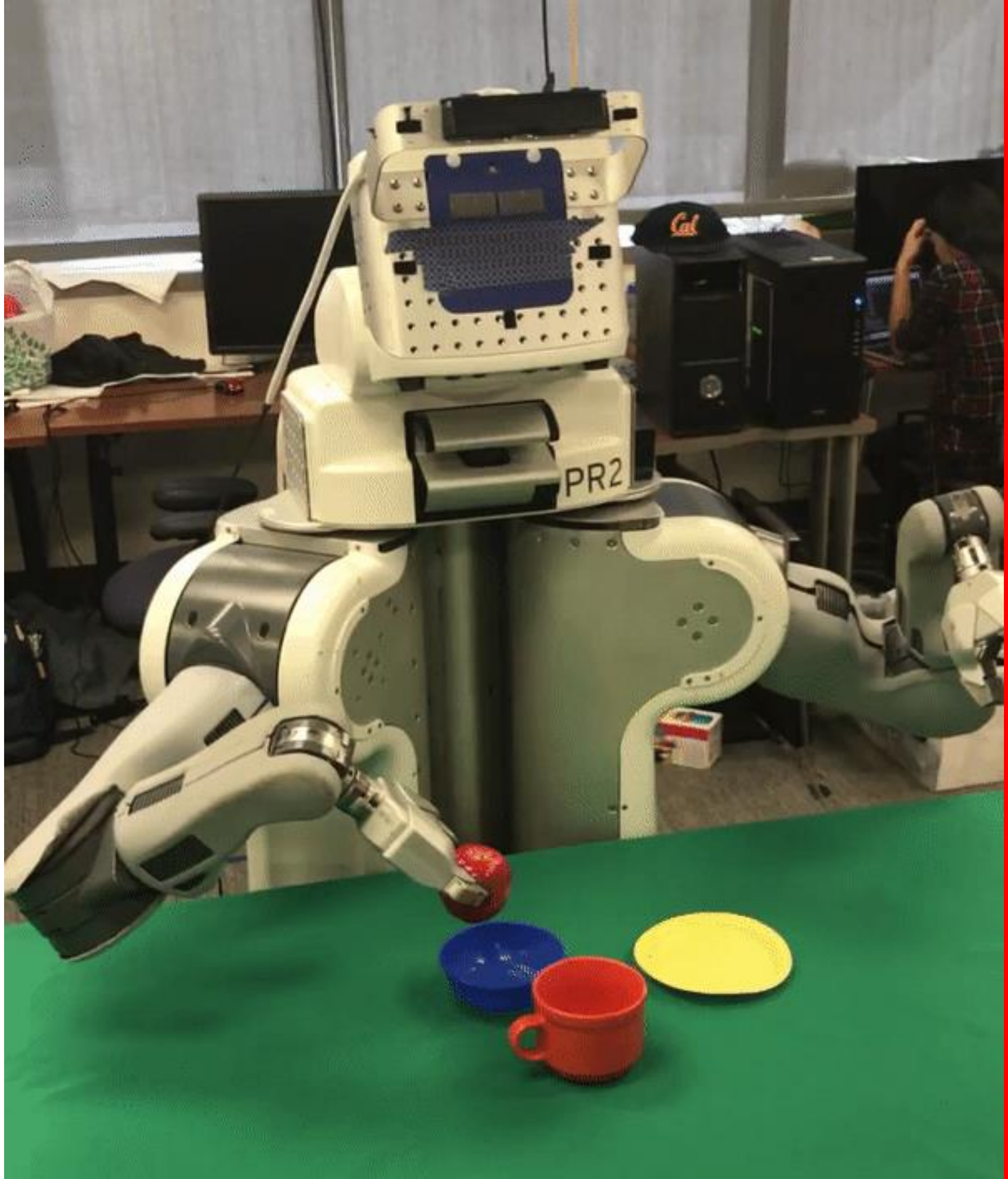
Skill Policies:

Imitation learning

What
is imitation learning?

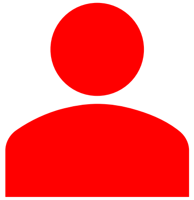
Subfield of ML where an agent (robot) learns to perform tasks by mimicking human demonstrations

[19]

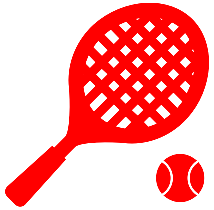


Why imitation learning?

bypass time-consuming exploration that would be required in a reinforcement learning setting



communicate user preferences for how a task ought to be done



describe concepts, such as a good tennis swing, that may be difficult to specify formally or programmatically

Types of Imitation Learning [1]

Behavior Cloning

Direct Policy Learning

Inverse Reinforcement Learning

Behavior Cloning

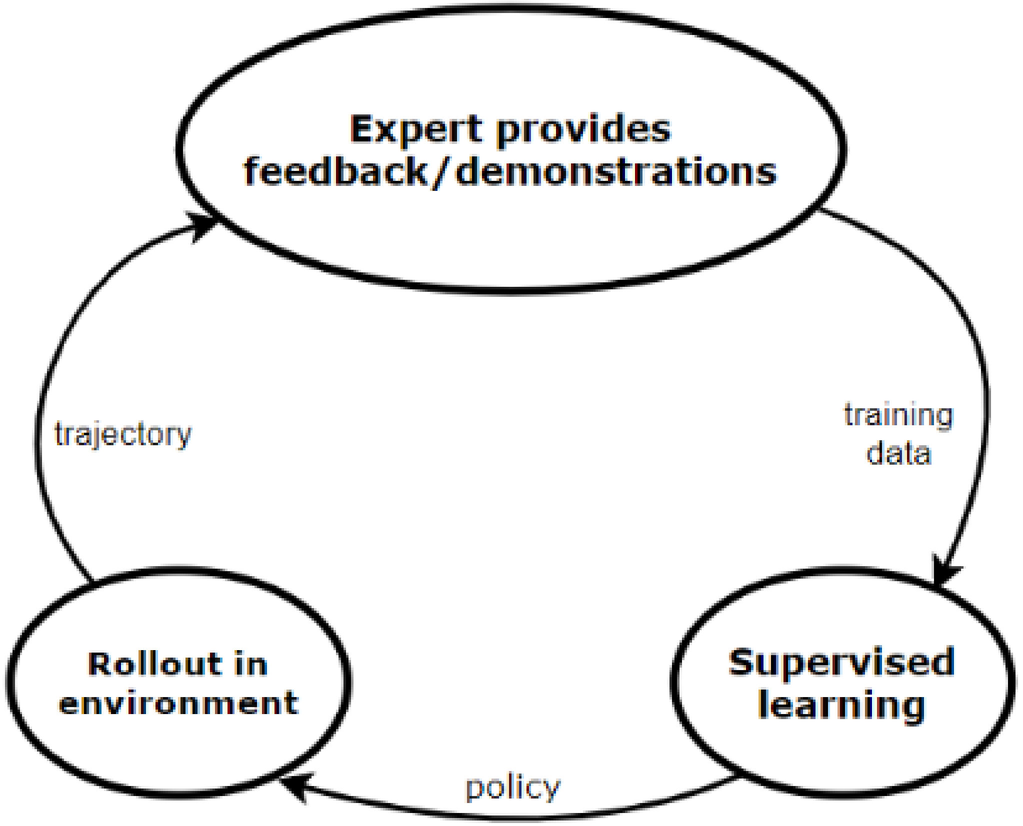
Behavioral Cloning is **Supervised Learning**

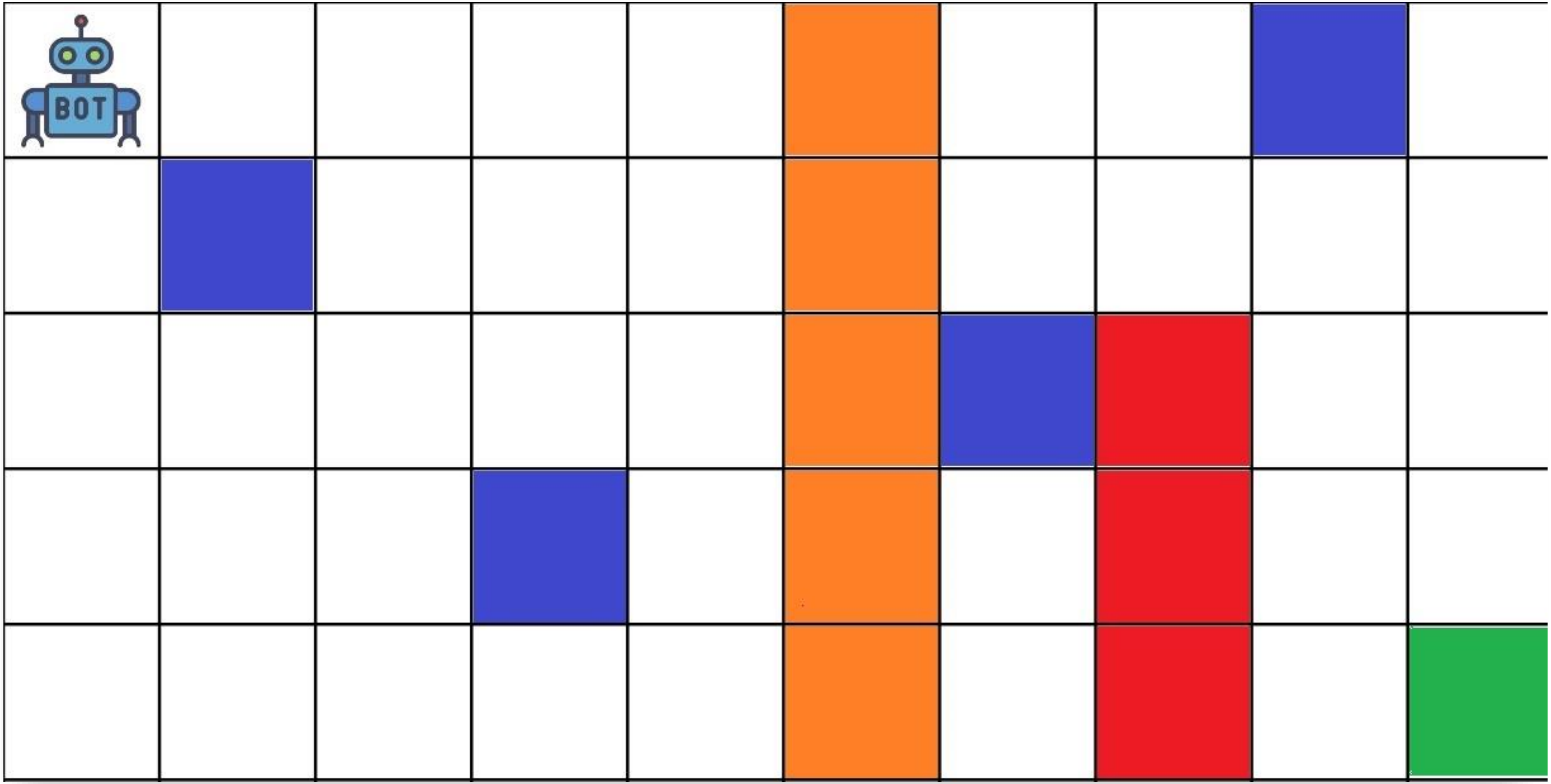
1. Collect demonstration data (state-action pairs)
2. Use state-action pairs as training data for supervised learning
3. Learn by minimizing the loss function

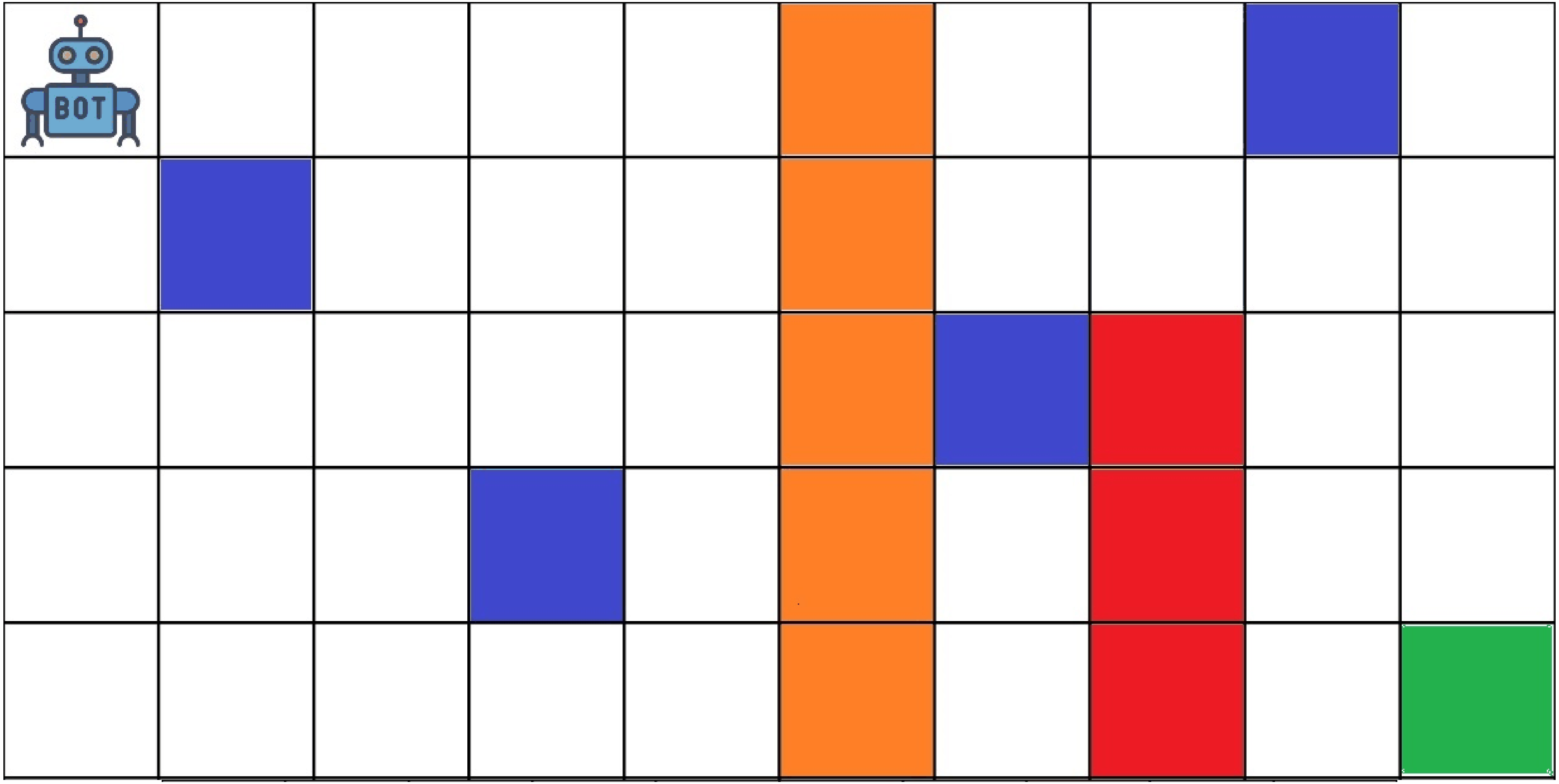
Behavior Cloning demo

Direct Policy Learning

- Improved version of behavior cloning
- Human expert needed during iterative training process
- Each training iteration, human corrects robot behavior
- New state-action pairs are used in the next training iteration




BOT


BOT

Inverse Reinforcement Learning

Attempt to infer the underlying reward function that the demonstrator was trying to optimize

1. Collect demonstrations of an expert performing a task.
2. Infer the underlying reward function that the expert is optimizing
3. Use reinforcement learning techniques to learn a policy that maximizes the inferred reward function.

Maximum Entropy

INITIALIZE REWARD WEIGHTS

COMPUTE POLICY

COMPUTE STATE VISITATION FREQUENCIES

UPDATE REWARD WEIGHTS

REPEAT

Skill Policies:
Skill Transfer

Parameterized Skills [2]

- In certain tasks, only some aspects of the context change, while other properties remain unchanged
- Manifold learning modulates policy parameters based on the changing task parameter

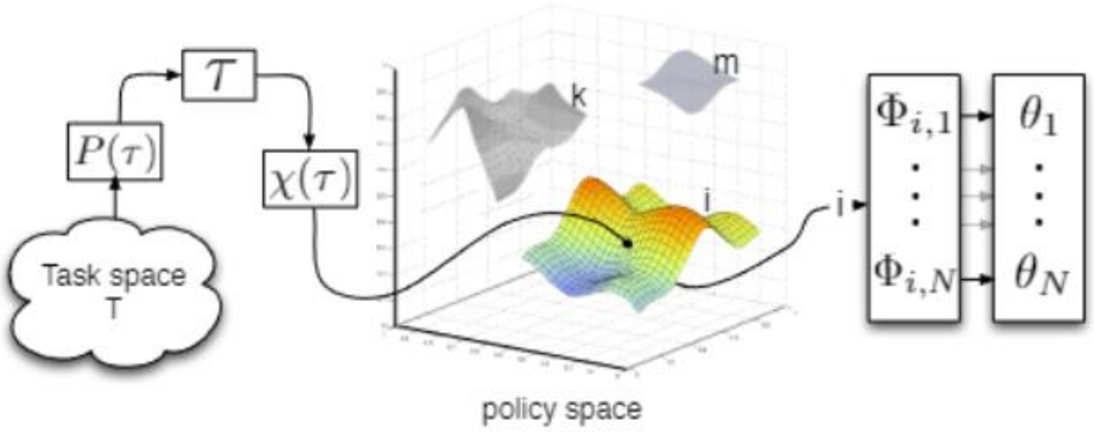


Figure 1. Steps involved in executing a parameterized skill: a task is drawn from the distribution P; the classifier χ identifies the manifold to which the policy for that task belongs; the corresponding regression models for that manifold map task parameters to policy parameters.

[17]

Metalearning [3]

- **"Learning to learn"**
- Model Agnostic Metalearning (MAML)
    - Given a sequence of tasks, the parameters of a given model are trained such that few iterations of gradient descent with few training data from a new task will lead to good generalization performance on that task. MAML **"trains the model to be easy to fine-tune."**

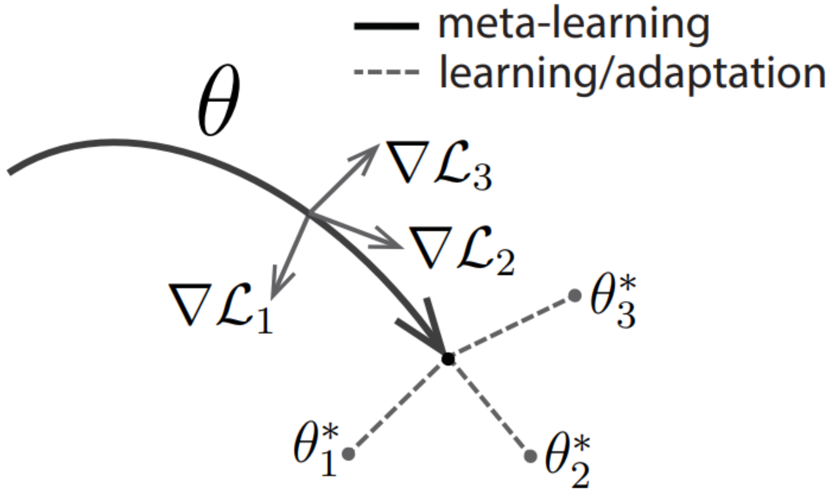
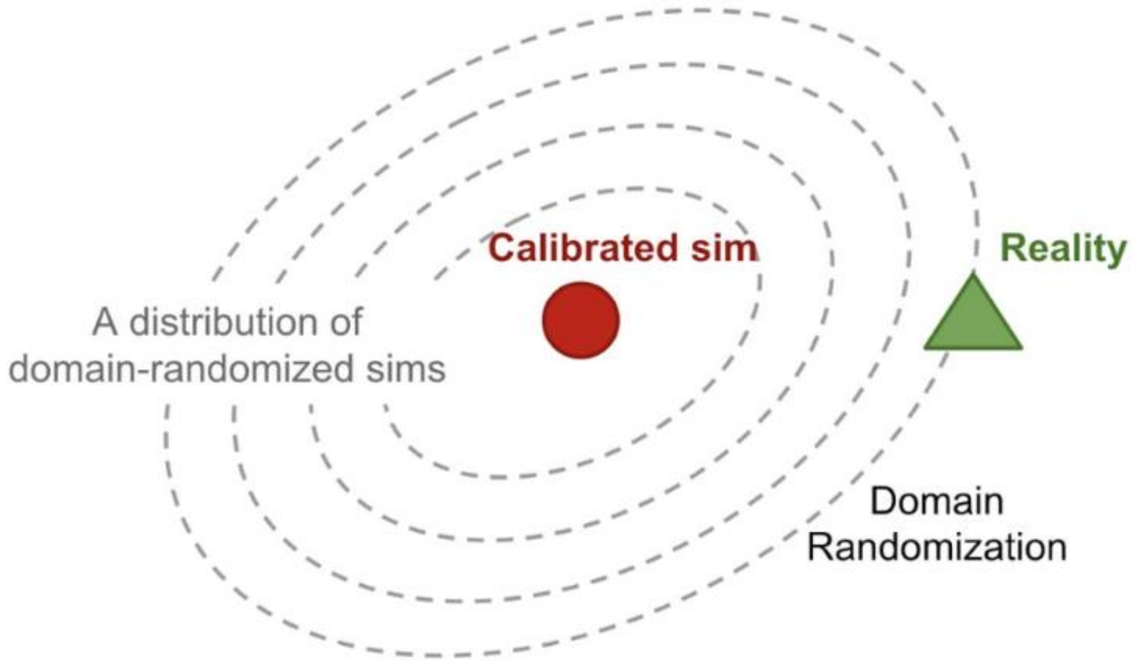


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

[18]

Domain Adaptation

- Used when two tasks are the **same at a high level** but **differ in low-level details**
- Example: sim2real task
- Popular method: **Domain Randomization**



[20]

Charecterizing skills by Preconditions and Effects

Pre- and Postconditions as Propositions and Predicates

Skill Monitoring and Outcome Detection

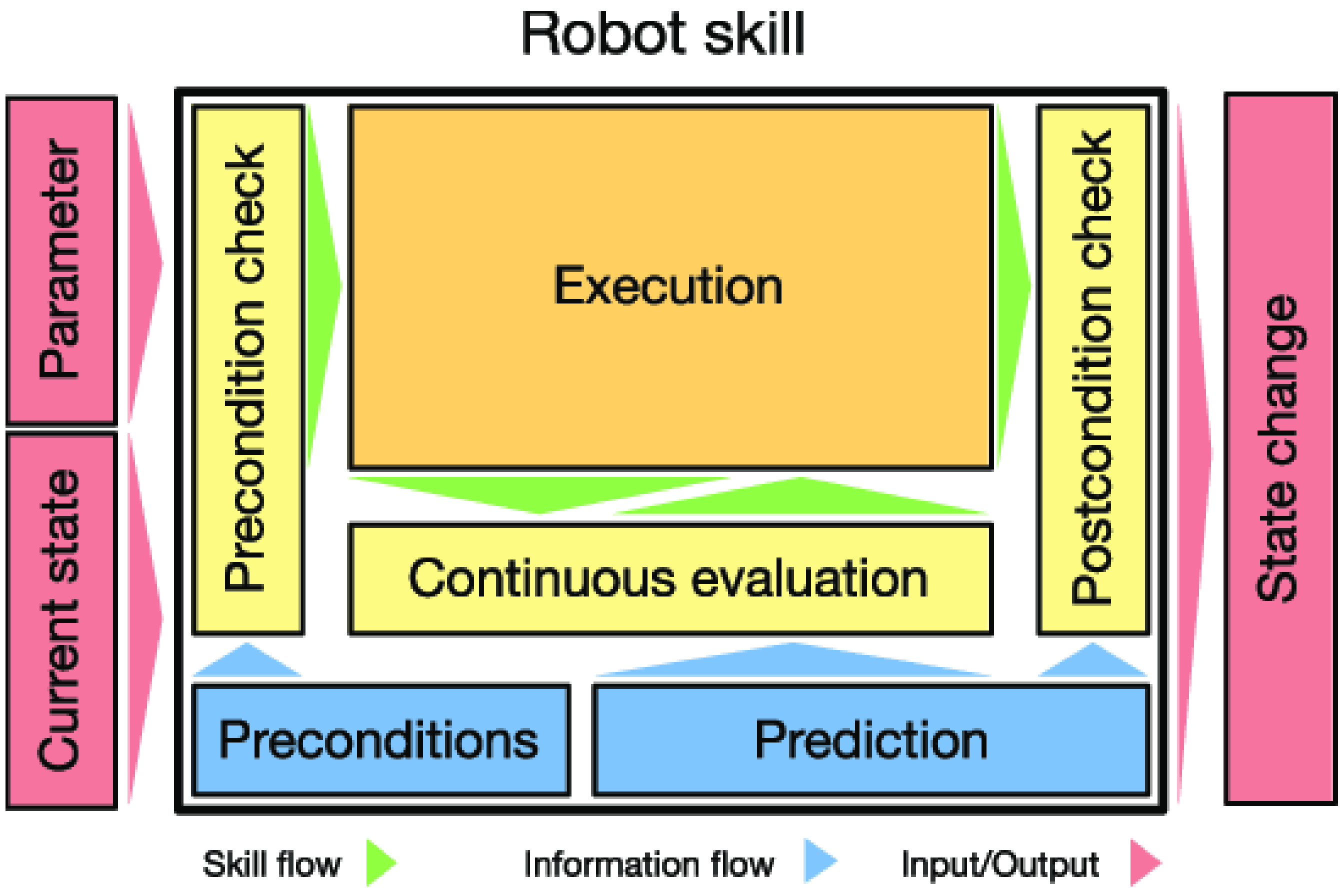
Learning Pre- and Postcondition Groundings

Predicates and Skill Synthesis

What are Pre- and Post- conditions?

- **Precondition**: A set of conditions that must be met for a robot to successfully execute a particular skill.

- **Postcondition**: A set of conditions describing the state of the environment after a robot has executed a particular skill.



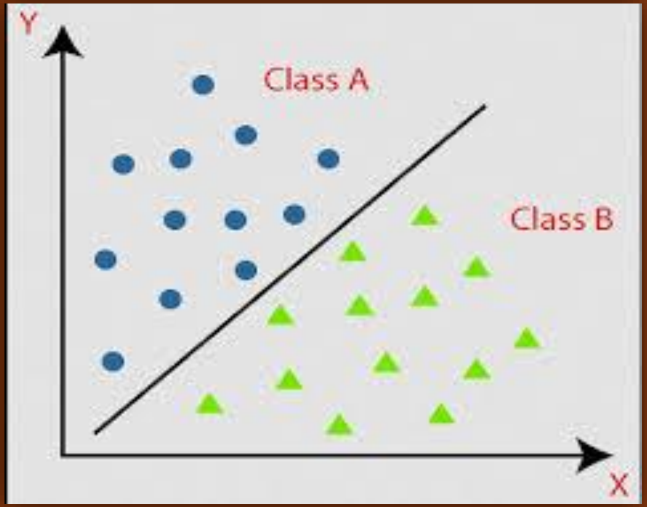
Pre- and Postconditions as Propositions and Predicates

Proposition	Predicate
AtTableB **GraspedKnifeAwRHand**	**At(TableB)** **Grasped(KnifeA, RHand)**

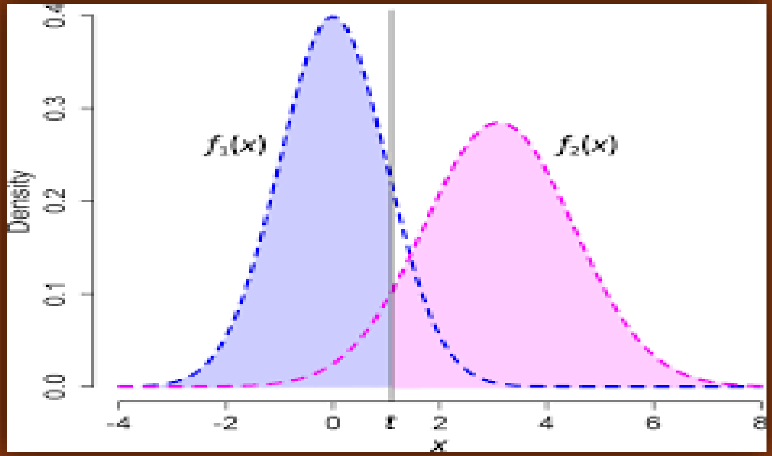
How can a robot map its complex environment to abstract predicates?

Grounding a Predicate is linking abstract symbols or logical expressions to real-world data and conditions.

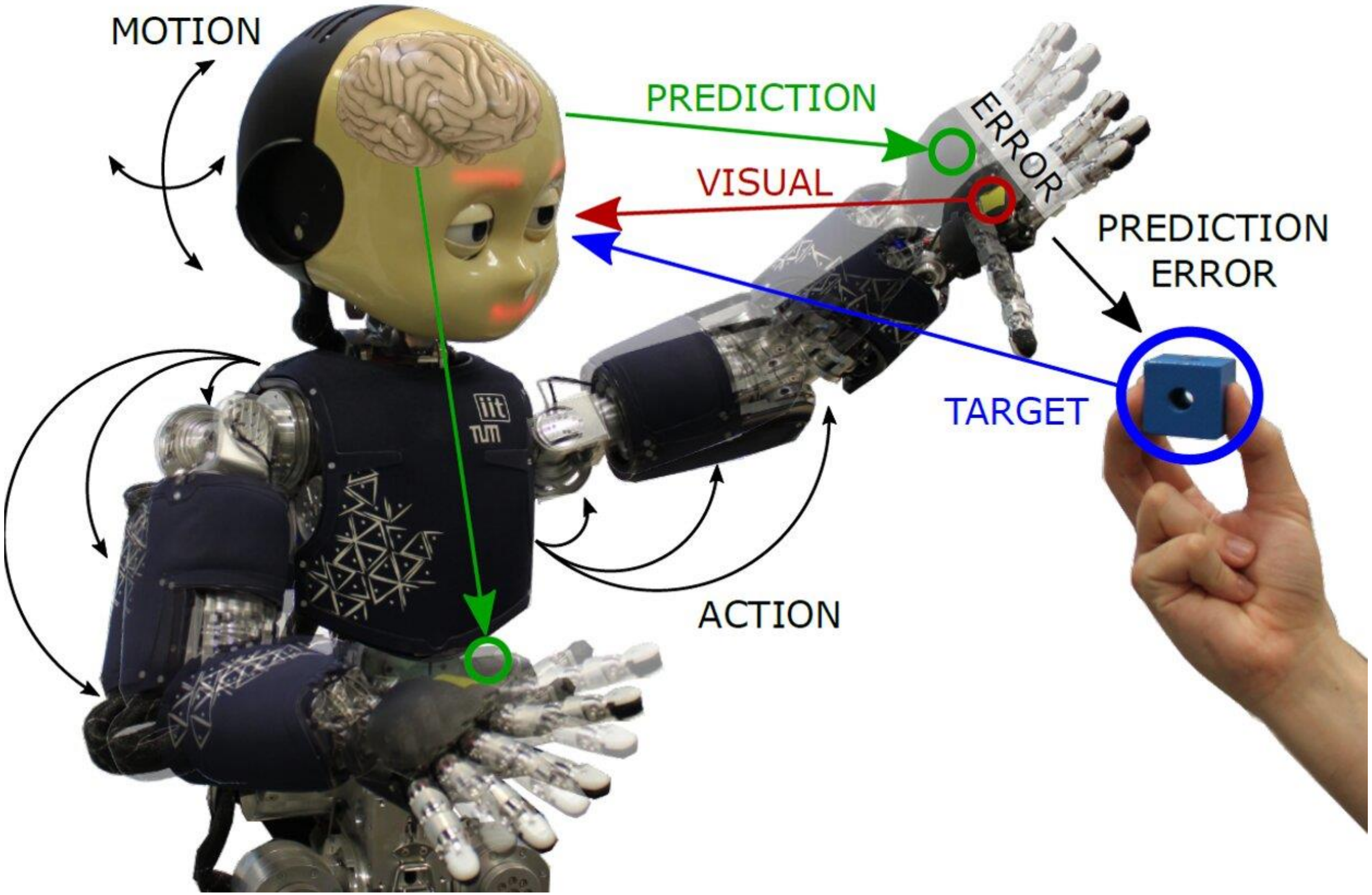
Classifier Representation:



Distribution Representation:



Ways to Monitor Outcomes and Errors



1. Learning Goal and Error Classifiers
2. Detecting Deviations from Nominal Sensory Values
3. Verifying Predicates

Compositional and Hierarchical task structures

Compositional and Hierarchical tasks

Structure of Robotic Skill Execution

Ways to Segment Trajectories into Component Skills

Skill Discovery While Solving Tasks

Complex tasks

Instruction 1:

Open the drawer.

Instruction 2:

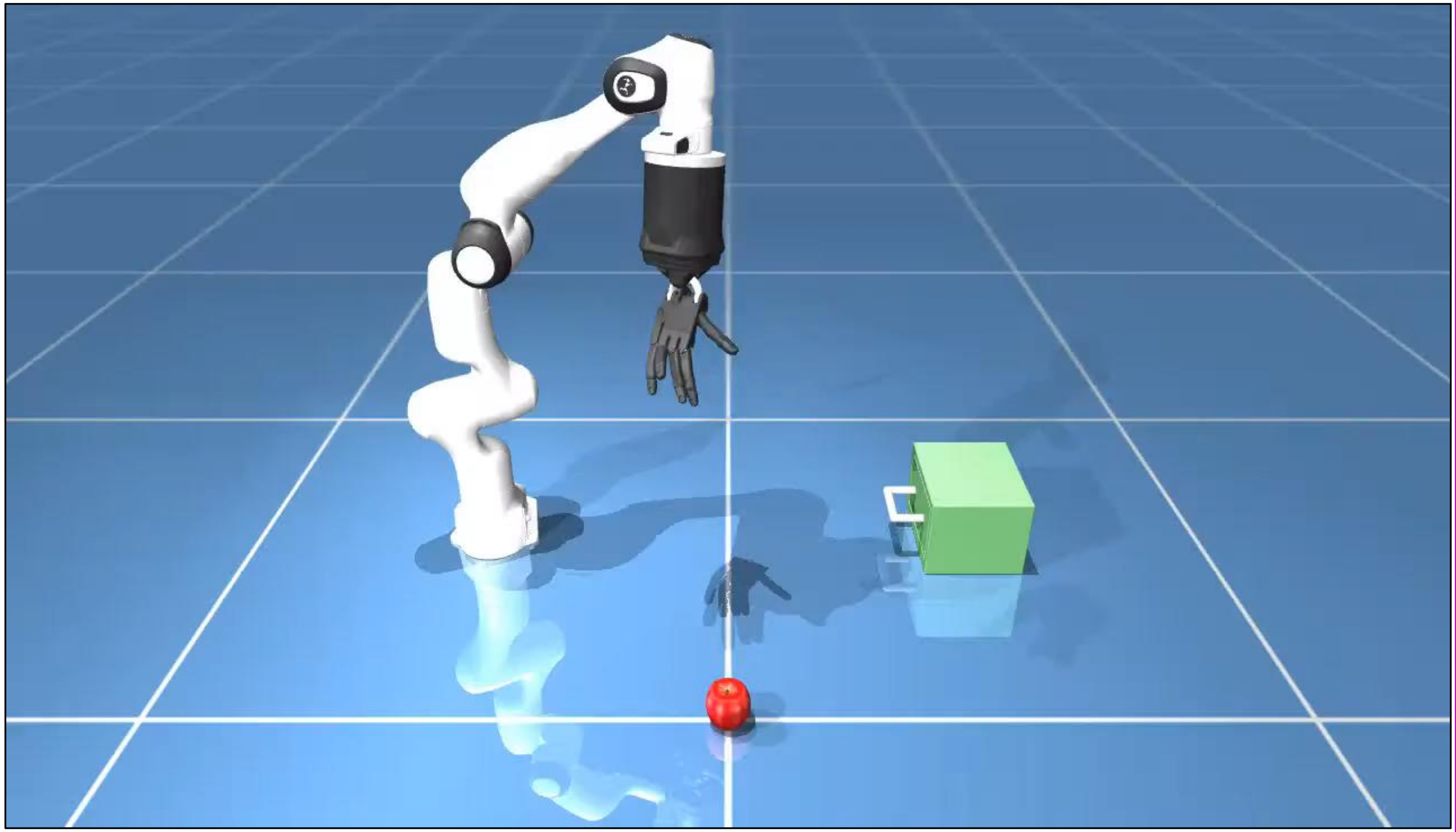
Put the apple inside the drawer while keep it open.

Instruction 3:

Release the apple and move hand away.

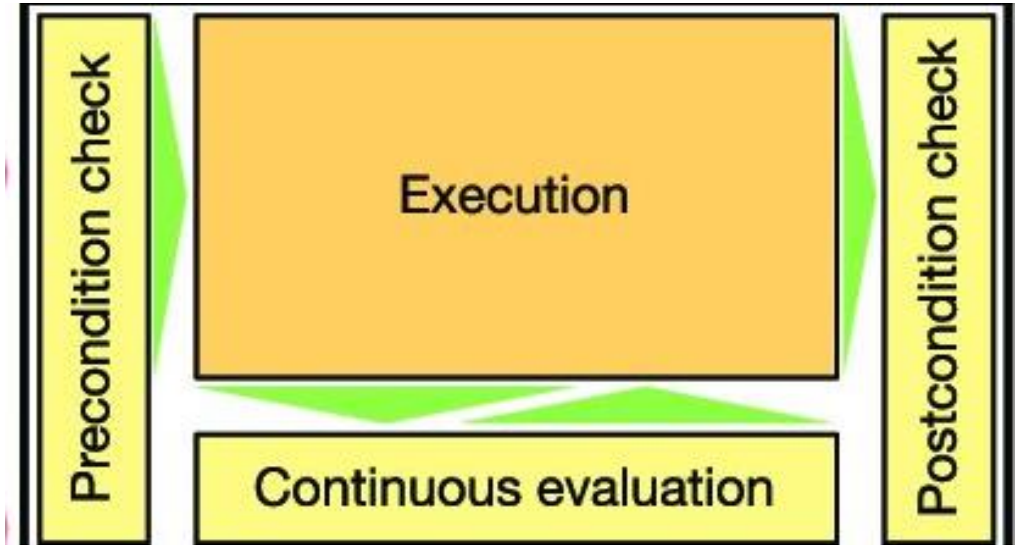
Instruction 4:

Close the drawer.



Structure of Robotic Skill Execution

$$o = (I_o, \beta_o, \pi_o),$$



- $I_o : S \to \{0, 1\}$ initiation set.
- $\beta_o : S \to [0, 1]$ termination condition.
- π_o option policy.

A **skill library** is a set of reusable skills for a robot.

How to identify skills?

Identifying Skills:

1. Discovering Skills While Solving Tasks

1 Task Execution	→	2 Event-Based Skill Creation	→	3 Skill Refinement

2. Segmenting Trajectories into Component Skills

Skill similarity	Specific Events	Language-guided

1. Segmentation Based on Skill Similarity

Ways to measure skill similarity

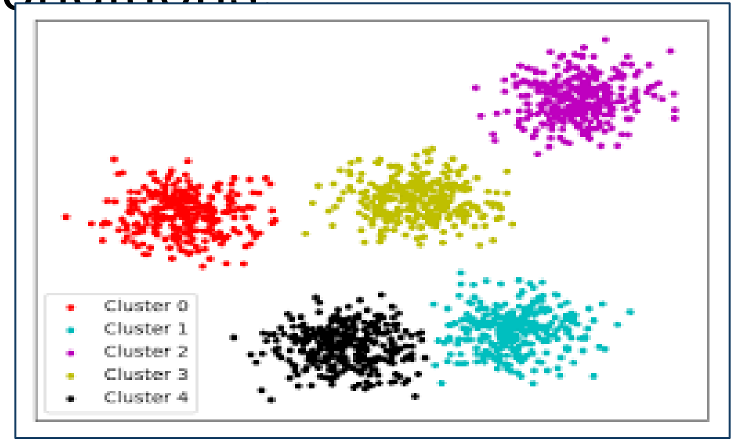
Policy Similarity:

- Measures skill similarity by fitting data to policies and evaluating distance in parameter space.

$$d^*(x,y) = \underbrace{\textsc{Dist}\big(\pi^*(x), \pi^*(y)\big)}_{\text{(A)}} + \underbrace{\gamma \mathcal{W}_1(d^*)\big(P^{\pi^*}(\cdot\,|\,x), P^{\pi^*}(\cdot\,|\,y)\big)}_{\text{(B)}} .$$

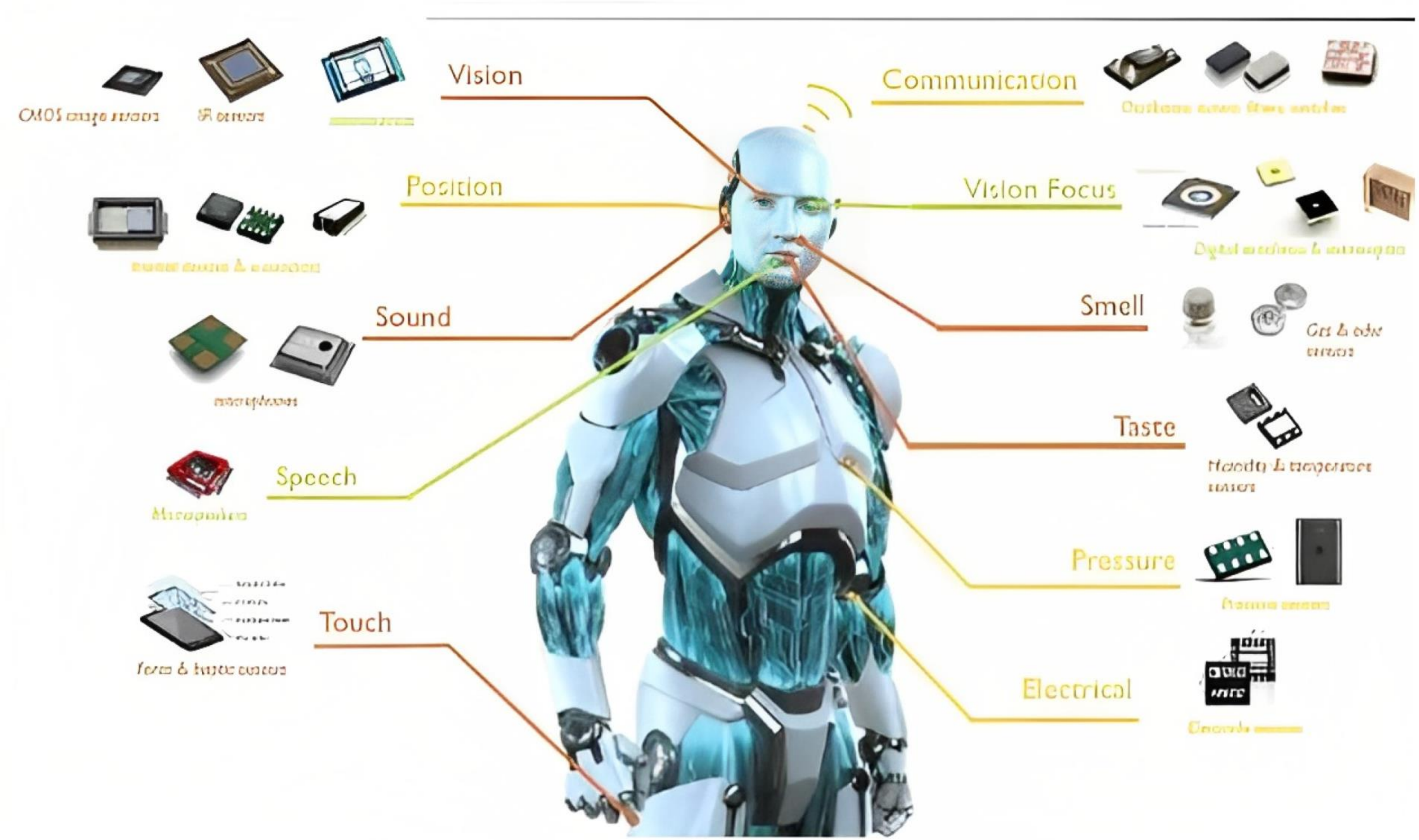
Pre- and Postcondition Similarity:

- Segments skills based on achieving goals from different initial conditions



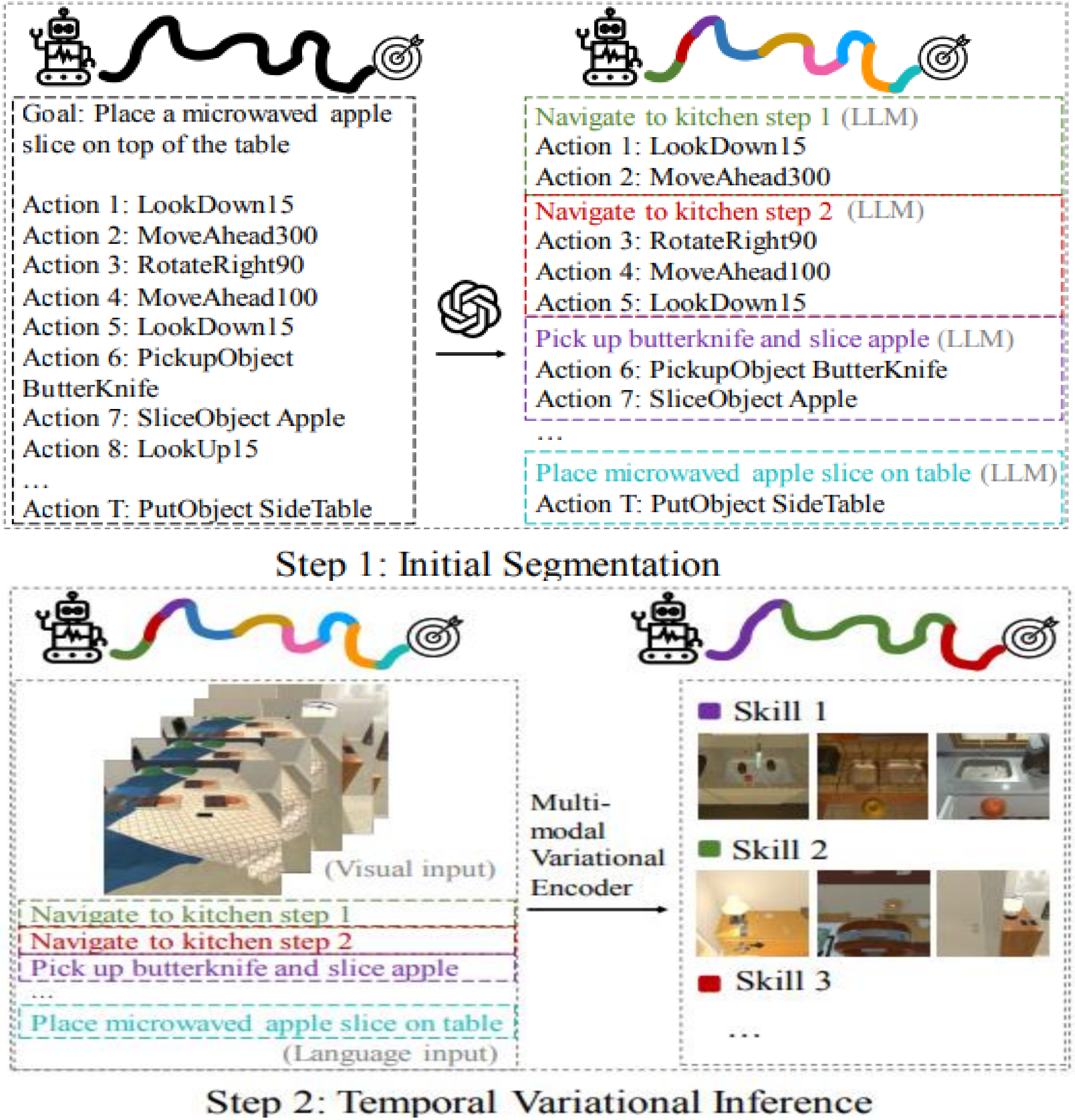
2. Segmentation Based on Specific Events

- Salient sensory events
- Transitioning between modes



Overall framework of LAST

(Language-guided Skill Learning with Temporal Variational Inference)



[10]

Language-guided Skill Learning with Temporal Variational Inference

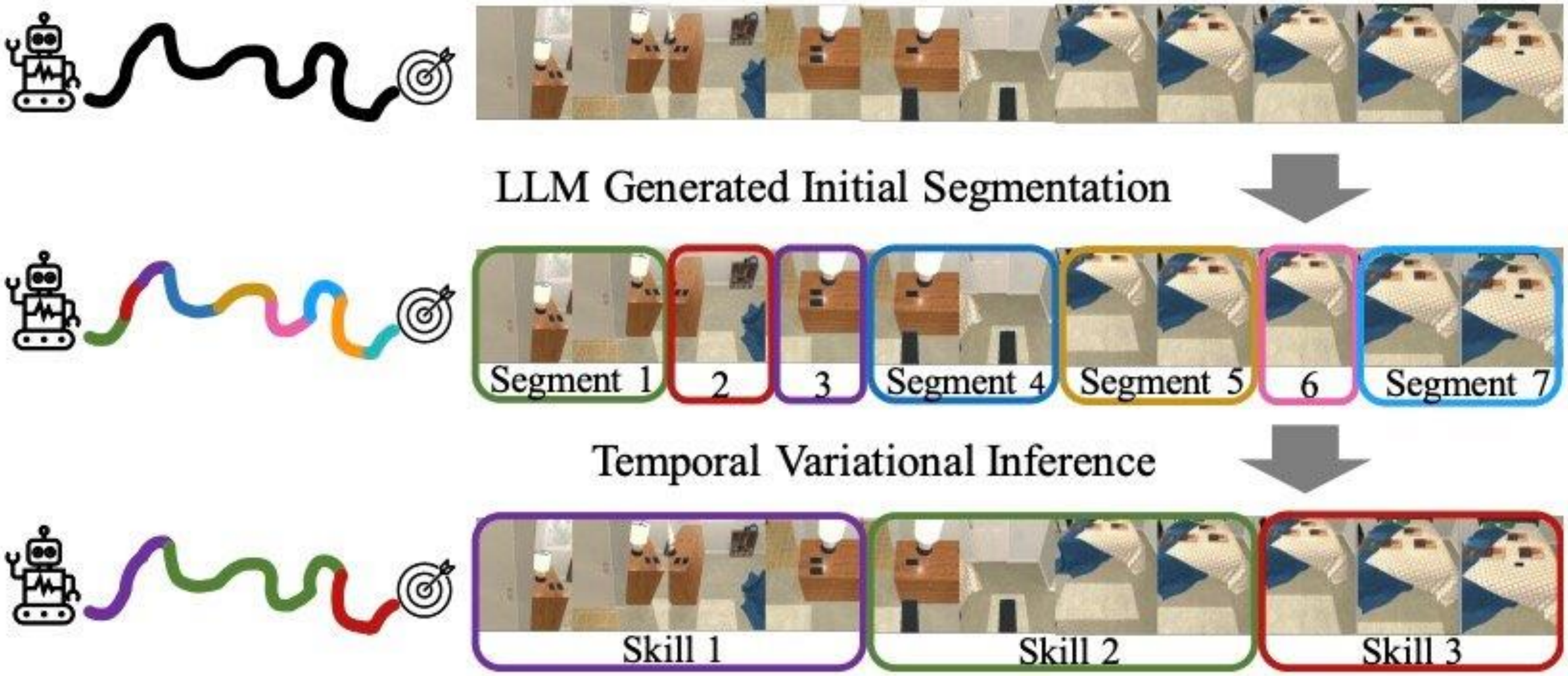
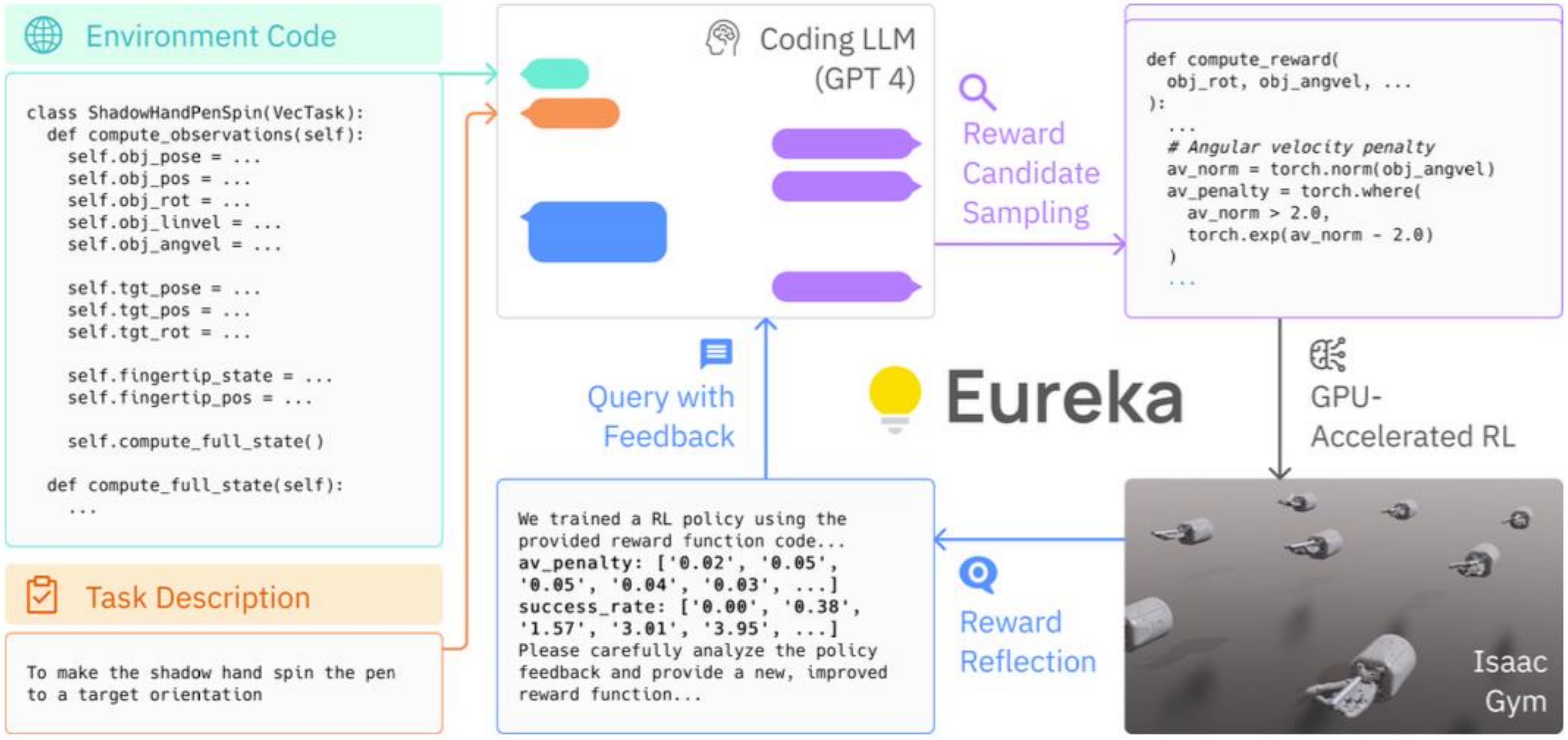


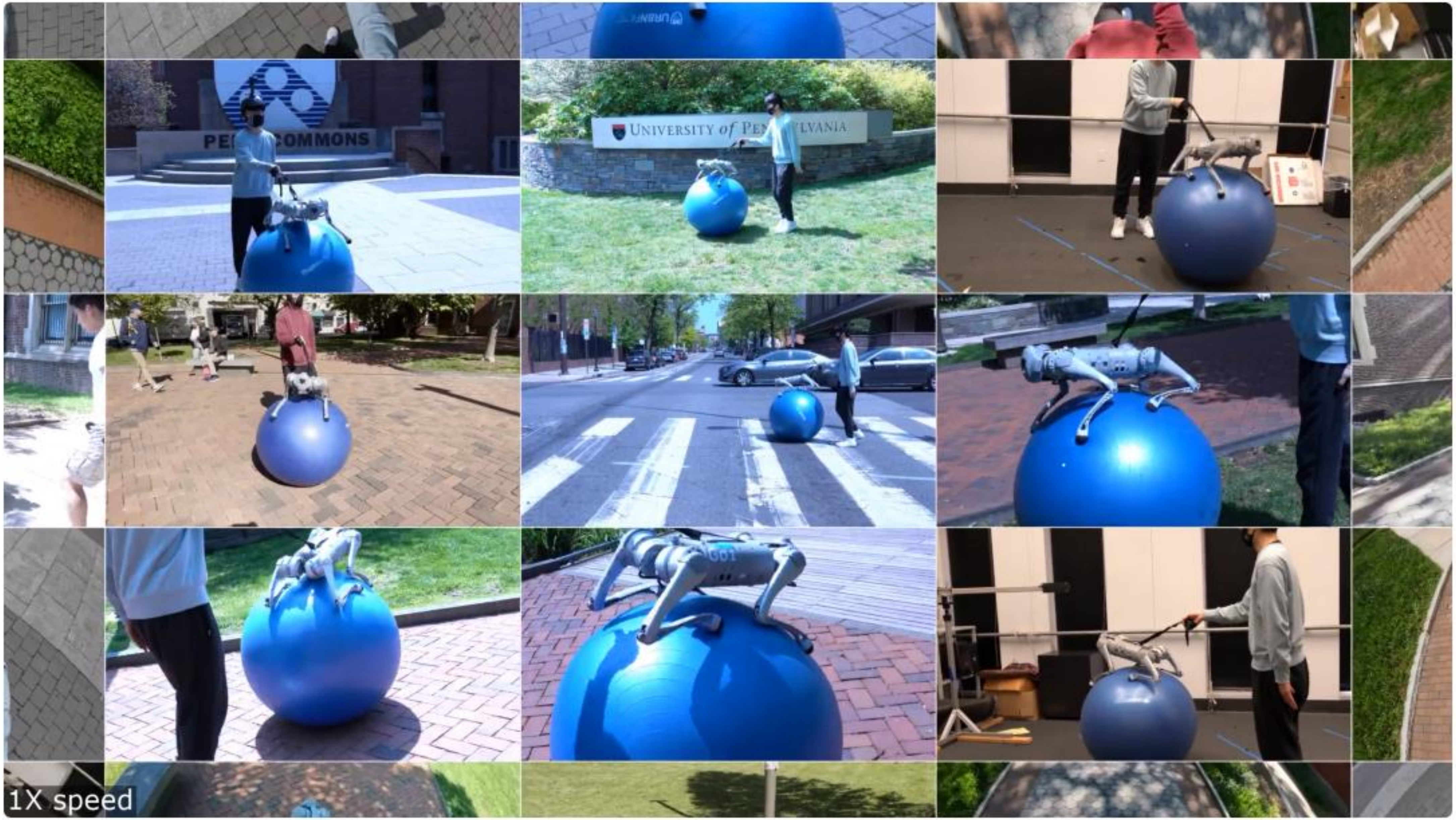
Figure 1. The trajectory segmentation and merging procedure.

[10]

Eureka and DrEureka

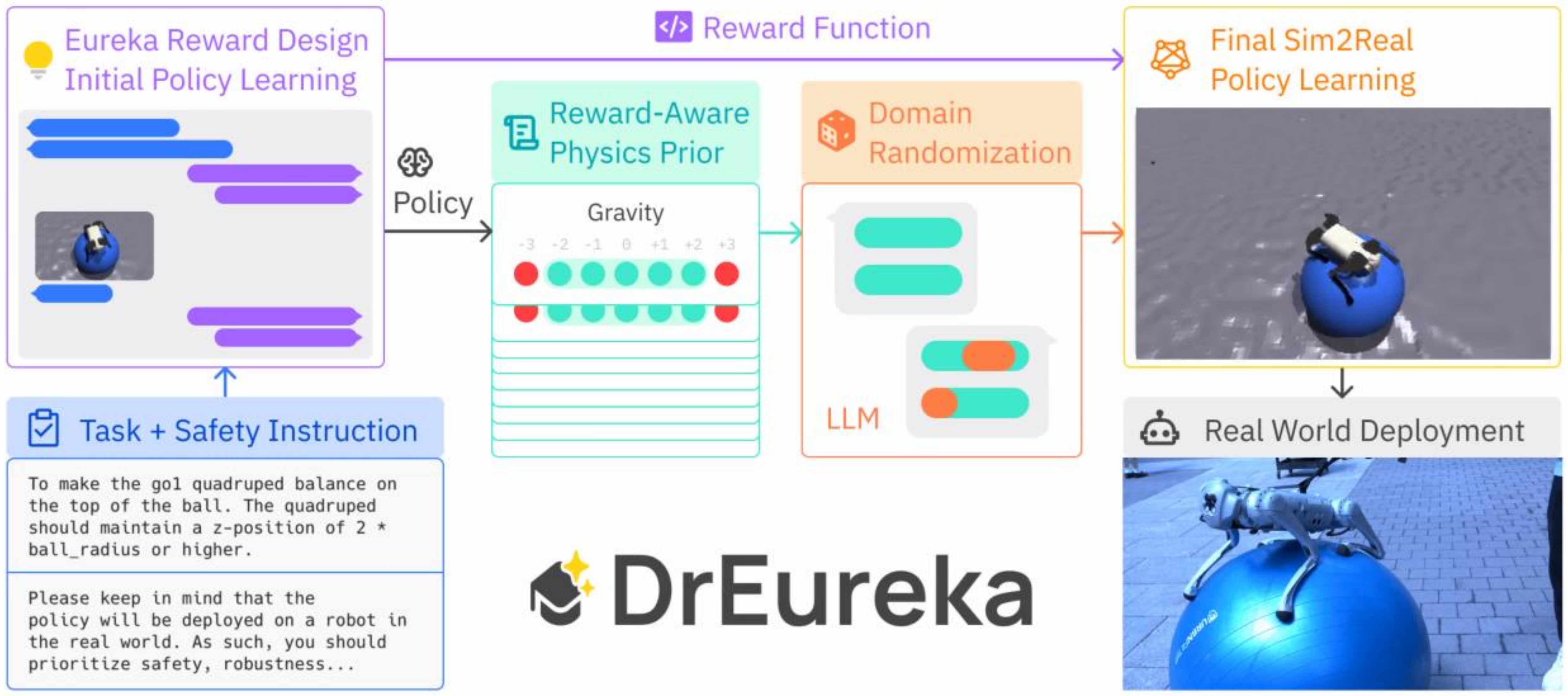

Environment Code
class ShadowHandPenSpin(VecTask):
def compute_observations(self):
self.obj_pose = ...
self.obj_pos = ...
self.obj_rot = ...
self.obj_linvel = ...
self.obj_angvel = ...
self.tgt_pose = ...
self.tgt_pos = ...
self.tgt_rot = ...
self.fingertip_state = ...
self.fingertip_pos = ...
self.compute_full_state()
def compute_full_state(self):
...
Task Description
To make the shadow hand spin the pen to a target orientation
Coding LLM (GPT 4)
Reward Candidate Sampling
def compute_reward(
obj_rot, obj_angvel, ...
):
...
# Angular velocity penalty
av_norm = torch.norm(obj_angvel)
av_penalty = torch.where(
av_norm > 2.0,
torch.exp(av_norm - 2.0)
)
...
Query with Feedback
Eureka
GPU-Accelerated RL
We trained a RL policy using the provided reward function code...
av_penalty: ['0.02', '0.05', '0.05', '0.04', '0.03', ...]
success_rate: ['0.00', '0.38', '1.57', '3.01', '3.95', ...]
Please carefully analyze the policy feedback and provide a new, improved reward function...
Reward Reflection
Isaac Gym

[11]



[12]

[12]

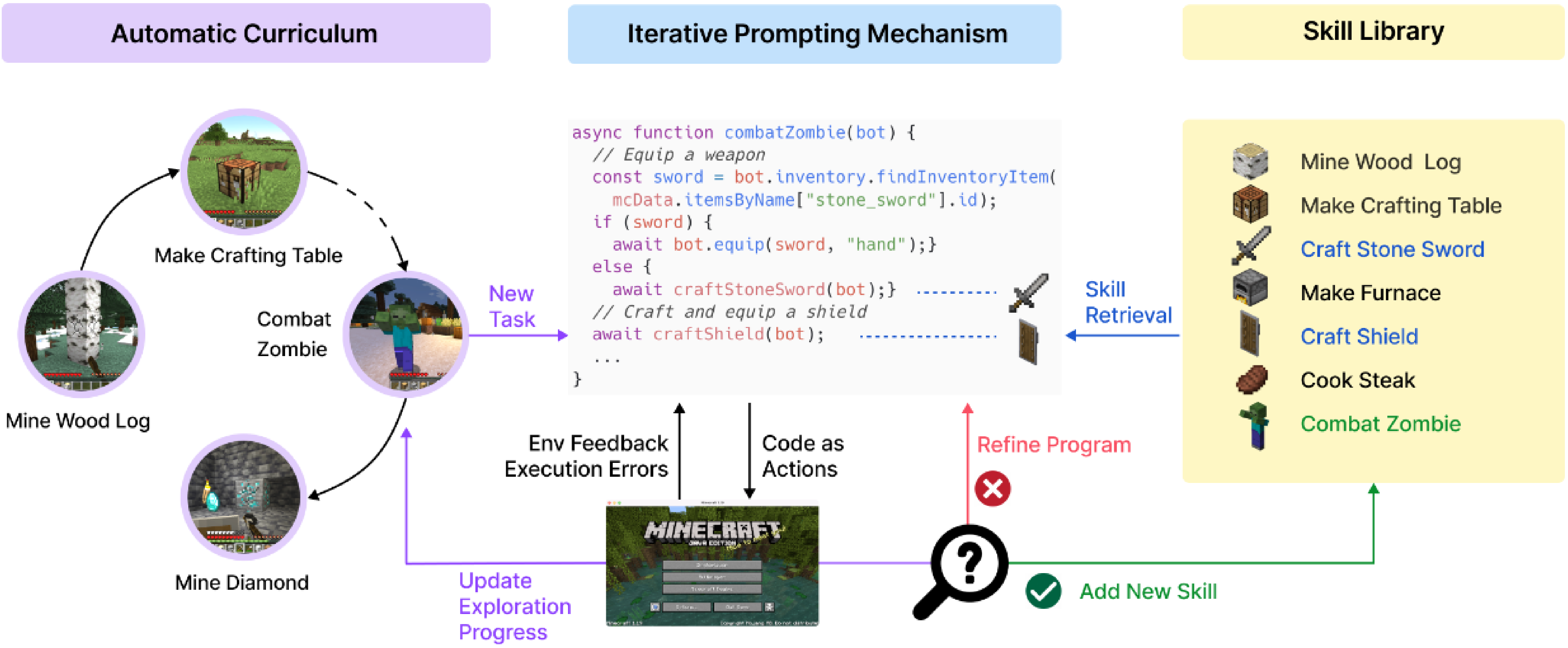

Eureka Reward Design
Initial Policy Learning
Reward Function
Final Sim2Real
Policy Learning
Policy
Reward-Aware
Physics Prior
Gravity
-3 -2 -1 0 +1 +2 +3
Domain
Randomization
LLM
Task + Safety Instruction
To make the go1 quadruped balance on the top of the ball. The quadruped should maintain a z-position of 2 * ball_radius or higher.
Please keep in mind that the policy will be deployed on a robot in the real world. As such, you should prioritize safety, robustness...
Real World Deployment
DrEureka

Voyager


Mine Diamond
Gather Cactus
Build Base
Fight Enderman
Build House
with Human Feedback
Mine Gold
Build Nether Portal
with Human Feedback
Hunt Pig
Catch Fish

[13]

Curiosity learning, curriculum learning, skills



[13]

References

[1] Kroemer, O., Niekum, S., & Konidaris, G. (2021). A review of robot learning for manipulation: Challenges, representations, and algorithms. Journal of machine learning research, 22(30), 1–82.

[2] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, & Pulkit Agrawal. (2022). Rapid Locomotion via Reinforcement Learning.

[3] Shixiang Gu, Ethan Holly, Timothy Lillicrap, & Sergey Levine. (2016). Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates.

[4] Benjamin Burchfiel and George Konidaris. Hybrid Bayesian eigenobjects: Combining linear subspace and deep network methods for 3D robot vision. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 6843–6850, 2018.

[5] Benjamin Burchfiel and George Konidaris. Bayesian eigenobjects: A unified framework for 3D robot perception. In Robotics: Science and Systems XIII, 2017.

[6] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic. Enhancing visual perception of shape through tactile glances. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3180–3186, 2013.

[7] Nuttapong Chentanez, Andrew G. Barto, and Satinder P. Singh. Intrinsically motivated reinforcement learning. In Advances in Neural Information Processing Systems 17, pages 1281–1288, 2005.

[8] Noémie Jaquier, Michael C. Welle, Andrej Gams, Kunpeng Yao, Bernardo Fichera, Aude Billard, Aleš Ude, Tamim Asfour, & Danica Kragic. (2024). Transfer Learning in Robotics: An Upcoming Breakthrough? A Review of Promises and Challenges.

[9] Schwarke, C., Klemm, V., Boon, M., Bjelonic, M., & Hutter, M. (2023). Curiosity-Driven Learning of Joint Locomotion and Manipulation Tasks. In Proceedings of The 7th Conference on Robot Learning (pp. 2594–2610). PMLR.

[10] Haotian Fu and Pratyusha Sharma and Elias Stengel-Eskin and George Konidaris and Nicolas Le Roux and Marc-Alexandre Côté and Xingdi Yuan (2024). Language-guided Skill Learning with Temporal Variational Inference.

[11] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, Anima Anandkumar (2023). Eureka: Human-Level Reward Design via Coding Large Language Models.

[12] Yecheng Jason Ma and William Liang and Hungju Wang and Sam Wang and Yuke Zhu and Linxi Fan and Osbert Bastani and Dinesh Jayaraman (2024). DrEureka: Language Model Guided Sim-To-Real Transfer. In book "Robotics: Science and Systems (RSS)".

[13] Guanzhi Wang and Yuqi Xie and Yunfan Jiang and Ajay Mandlekar and Chaowei Xiao and Yuke Zhu and Linxi Fan and Anima Anandkumar (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models.

[14] Yu, Wenhao and Gileadi, Nimrod and Fu, Chuyuan and Kirmani, Sean and Lee, Kuang-Huei and Gonzalez Arenas, Montse and Lewis Chiang, Hao-Tien and Erez, Tom and Hasenclever, Leonard and Humplik, Jan and Ichter, Brian and Xiao, Ted and Xu, Peng and Zeng, Andy and Zhang, Tingnan and Heess, Nicolas and Sadigh, Dorsa and Tan, Jie and Tassa, Yuval and Xia, Fei (2023). Language to Rewards for Robotic Skill Synthesis.

[15] Kober, J. (2020). Robot Learning. In: Baillieul, J., Samad, T. (eds) Encyclopedia of Systems and Control. Springer, London. https://doi.org/10.1007/978-1-4471-5102-9_100027-1

[16] Rishabh Agarwal and Marlos C. Machado and Pablo Samuel Castro and Marc G. Bellemare (2021). Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning.

[17] Da Silva, B., Konidaris, G., & Barto, A. G. (2012). Learning Parameterized Skills. Autonomous Learning Laboratory, Computer Science Department, University of Massachusetts Amherst; MIT Computer Science and Artificial Intelligence Laboratory.

[18] Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the International Conference on Machine Learning (ICML).

[19] https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c

[20] https://www.researchgate.net/figure/An-illustration-of-the-difference-between-aleatoric-and-epistemic-uncertainties-The-dots_fig1_368244388

[21] https://youtu.be/7Er84QulUBs?si=6ne4hN9kH7akd897

[22] https://images.app.goo.gl/7R5wD4bKJLeG5ZGP7

[23] https://www.frontiersin.org/files/Articles/534750/frobt-07-00082-HTML-r1/image_m/frobt-07-00082-g001.jpg