Survey: Cross-Embodiment Manipulation Policy

Zhuoheng Li



A policy

For any task and any robot

A policy

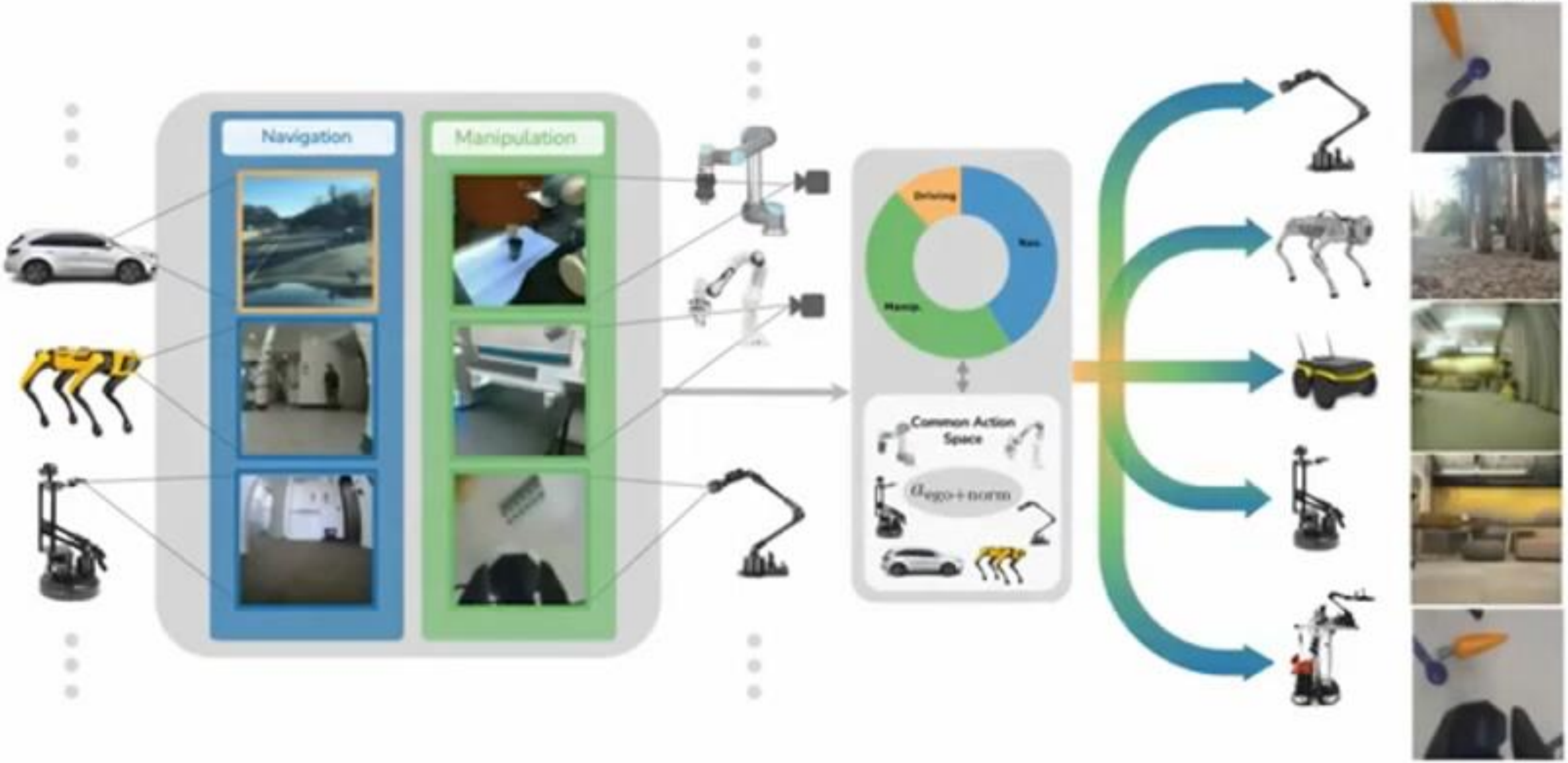
For **any task** and any robot

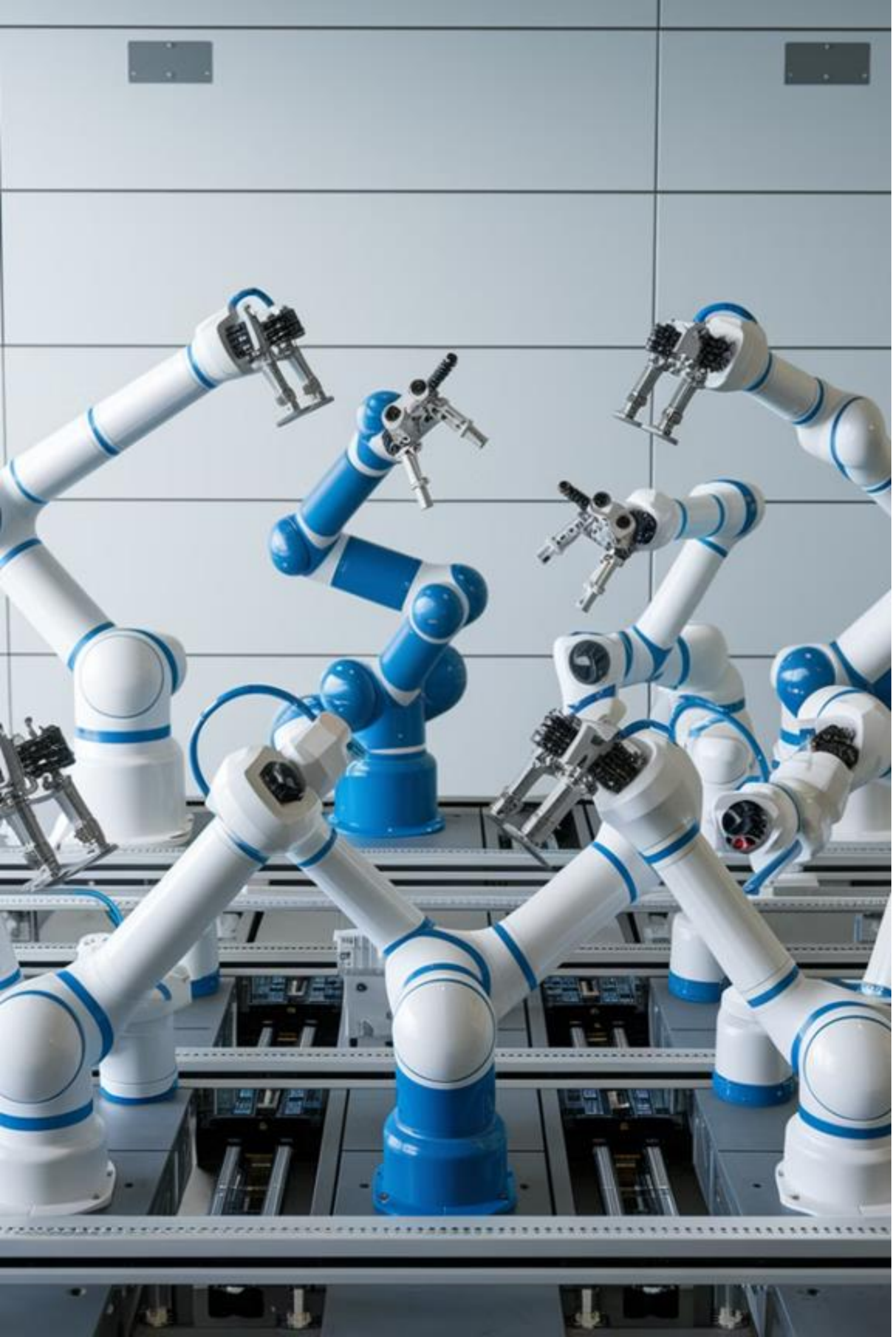

autonomous, 1x speed
π

A policy

For any task and **any robot**

Single Model Controlling All Robots





Why “Cross-Embodiment”?

One Controls All

Control different robotic arms with a single neural network

Use Less data from new embodiment

After pretraining of the cross-embodiment

Absorb Knowledge from Open-Source Datasets

which has different arm embodiments, camera perspectives, and gripper types.

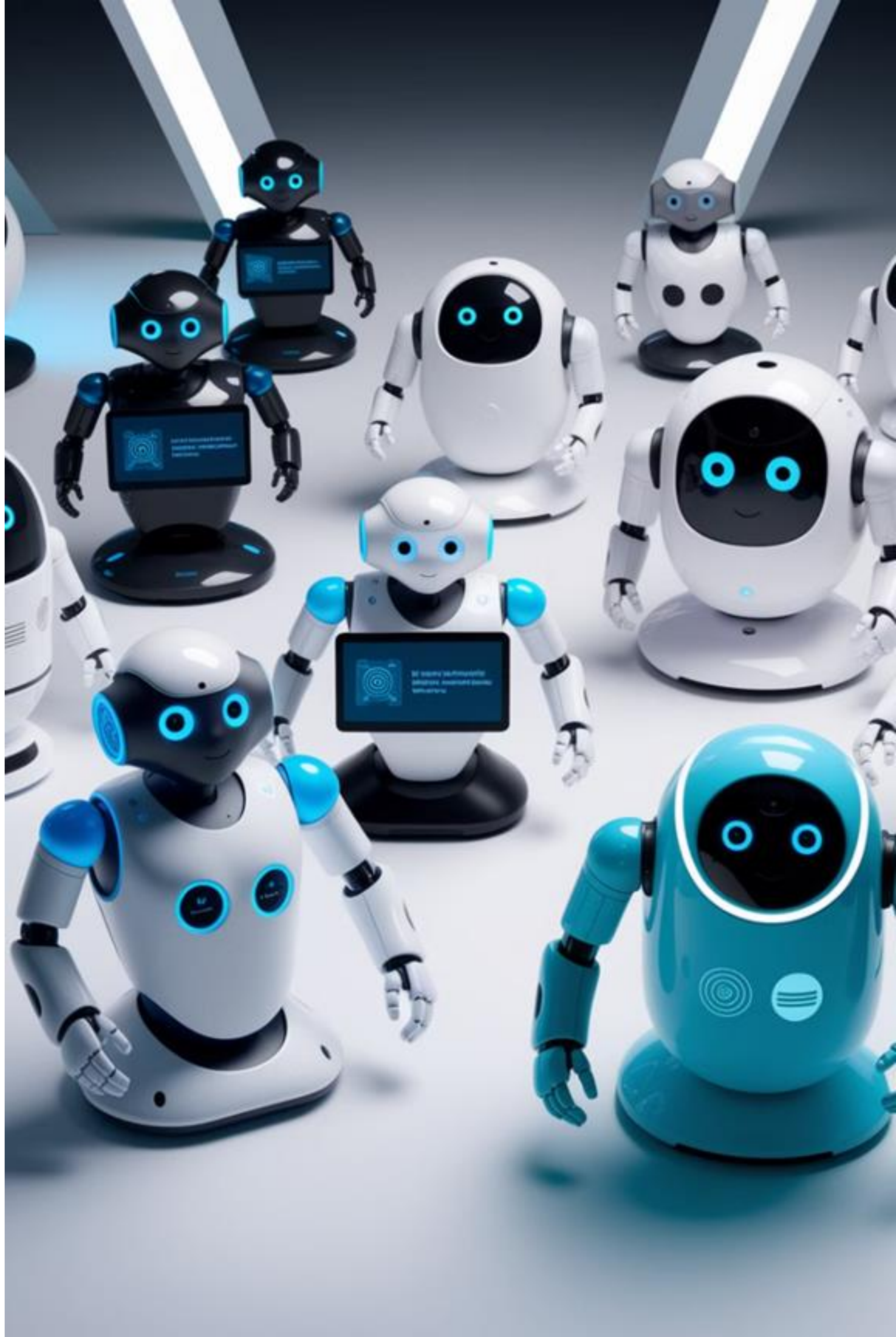
Two stages

Pretraining

Large multi-robot dataset from source domain

Finetuning

Small dataset from target robot



Two metrics

Success Metrics

Higher success with less data

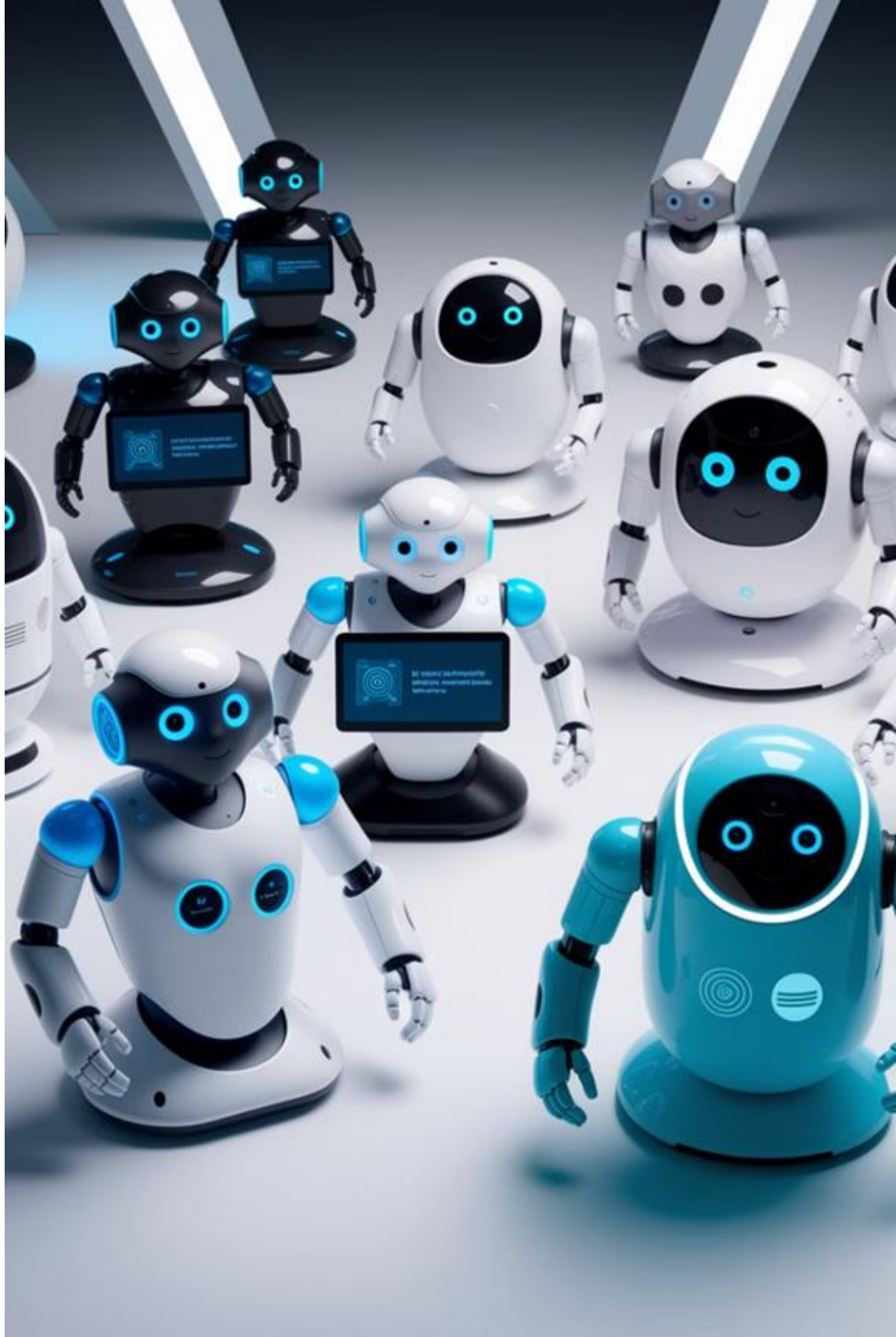
Retention of pre-trained tasks

Key Challenges

Preventing catastrophic forgetting

Managing embodiment differences

Optimizing data efficiency



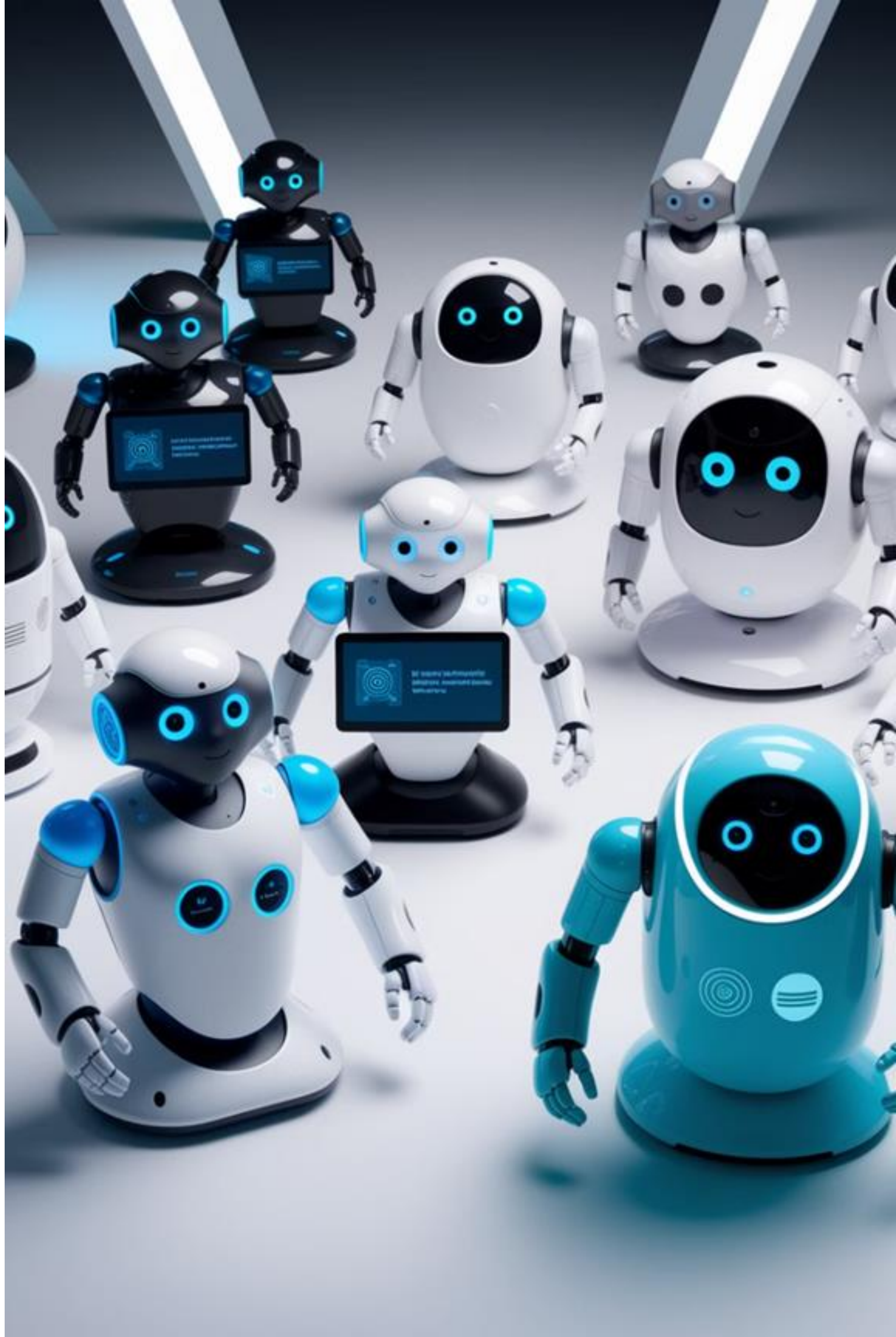
Two approaches

Embodiment-Agnostic Policy

Make the input / output / internal representations similar for different robots

Embodiment-Aware Policy

Use the robotics arm geometric configuration / camera view perspectives as conditional input



Embodiment-Agnostic Policy

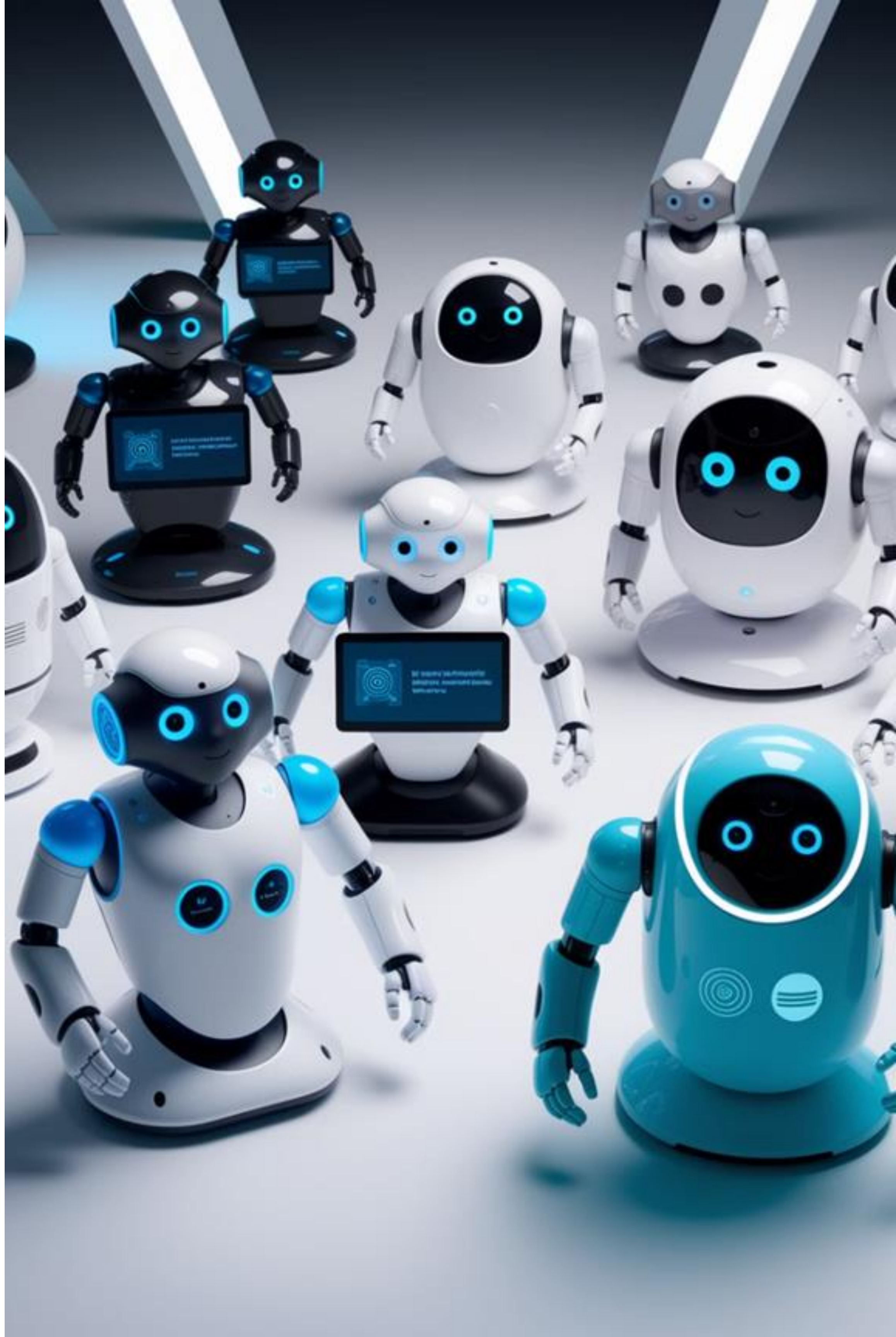
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- Source domain manipulation

2 Few-Shot Transfer Approaches

- Domain-invariant transitions
- Invariant features
- Hierarchical policies
- I don’t care, just train a large VLA



Embodiment-Agnostic Policy

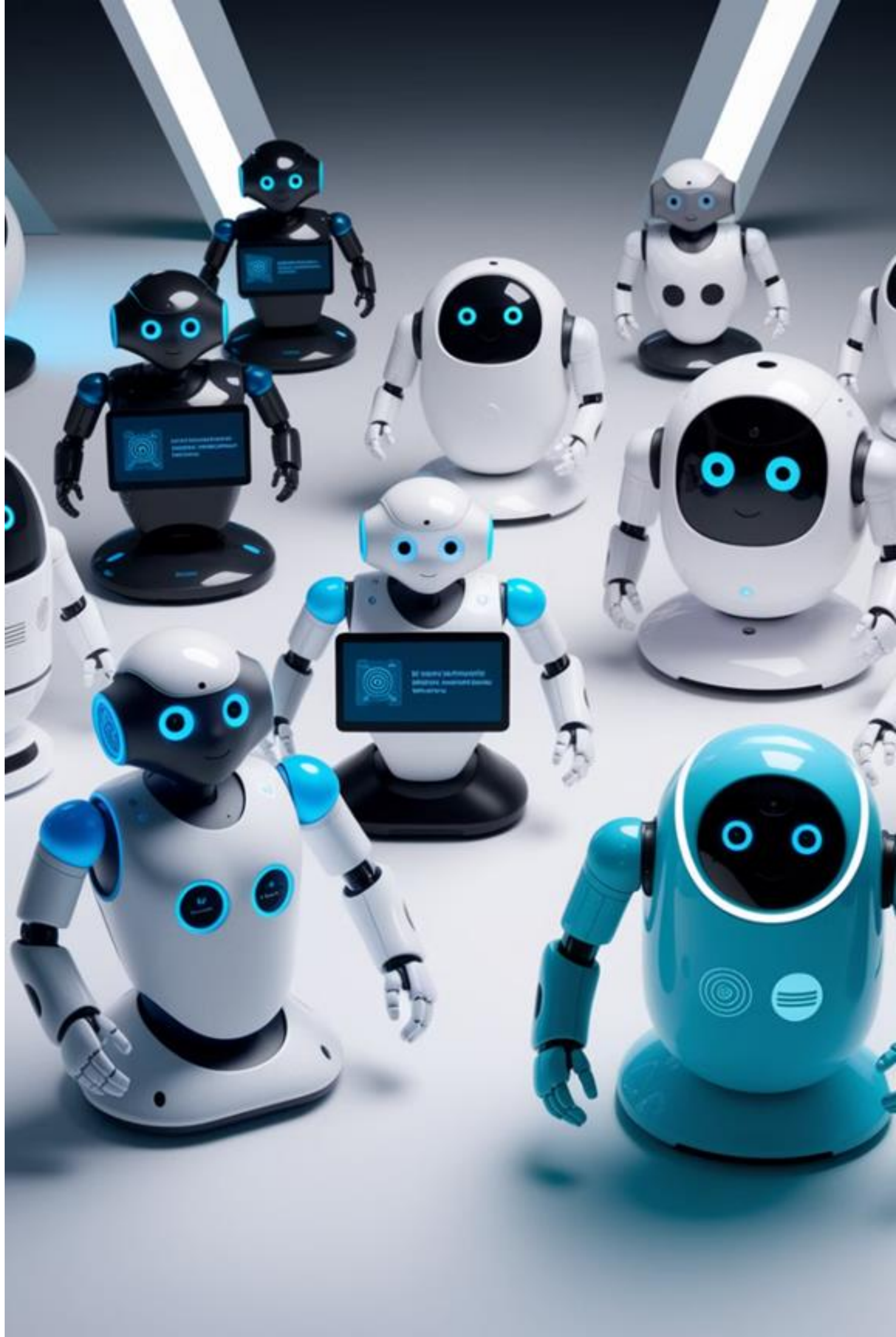
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- **Source domain manipulation**

2 Few-Shot Transfer Approaches

- Domain-invariant transitions
- Invariant features
- Hierarchical policies
- I don’t care, just train a large VLA



Source Domain Manipulation: Unifying Model Input & Output Across Embodiments

You can either do…

Special Hardware Design

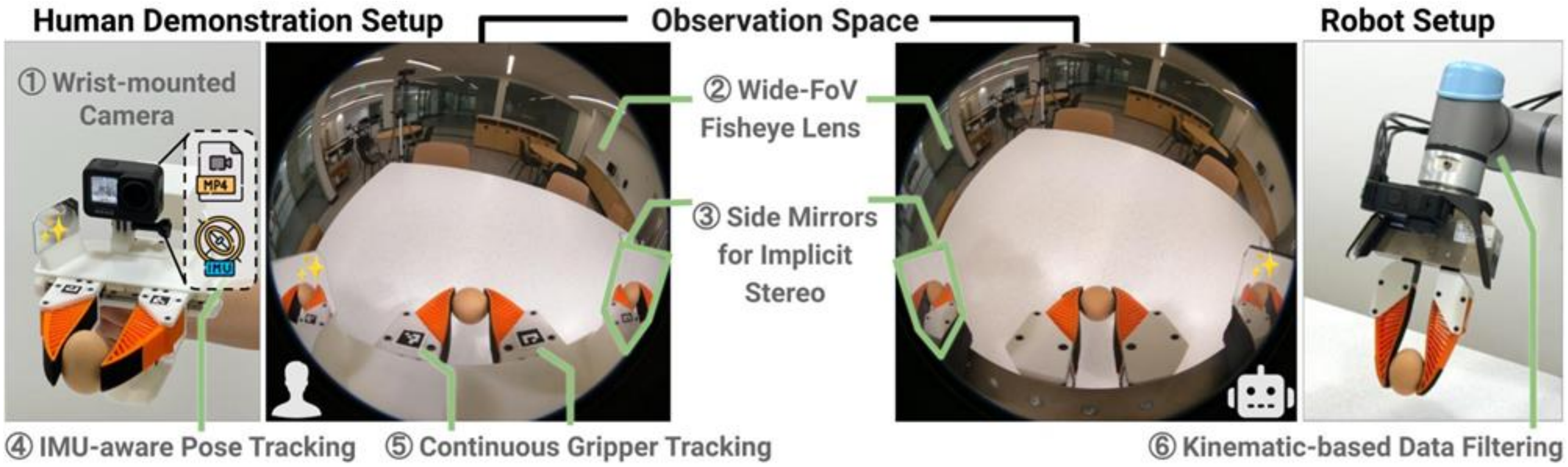


Image Editing

UMI: Data Collection

UMI: Deployment

Hardware-Based Solutions - Approach

Concept

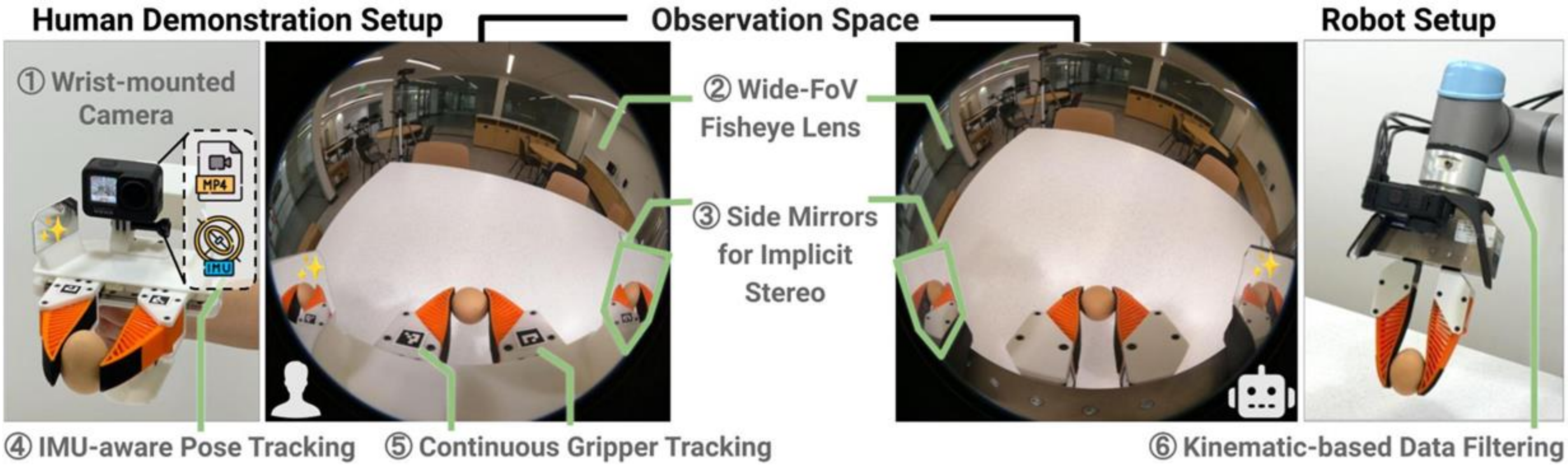
Use specialized hardware to standardize interfaces across different robotic arms.

This approach creates physical consistency between platforms.

Example Implementation

UMI employs hand-held grippers to unify control across different robotic arm systems.

The hardware creates a common physical interface for all arms.



Chi, C., Xu, Z., Pan, C., Cousineau, E., Burchfiel, B., Feng, S., ... & Song, S. (2024). Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*.

UMI: How it works?

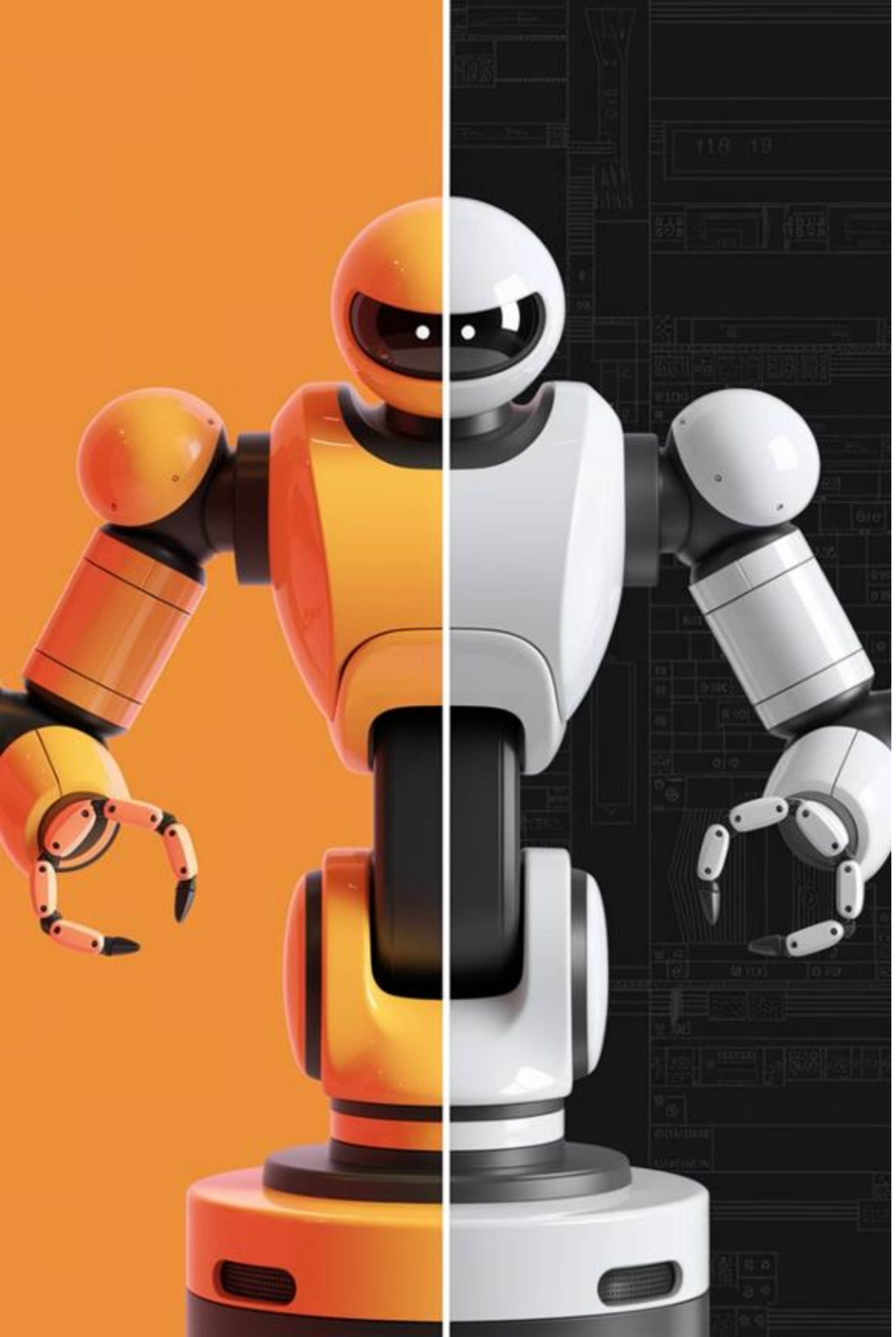


Image Editing - Key Papers

Mirage

Inpaint images at test time **to a single embodiment** with robot masks, physical simulator, and **rule-based fast matching inpainting** algorithm.

LookMaNoHand

Finetune **an SVD to remove franka panda robot** from the images.

RoviAug

Finetune an **SVD to inpaint robot A to robot B.**

Chen, L. Y., Hari, K., Dharmarajan, K., Xu, C., Vuong, Q., & Goldberg, K. (2024). Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *arXiv preprint arXiv:2402.19249*.

Chang, M., Prakash, A., & Gupta, S. (2023). Look ma, no hands! agent-environment factorization of egocentric videos. *Advances in Neural Information Processing Systems*, *36*, 21466-21486.

Chen, L. Y., Xu, C., Dharmarajan, K., Irshad, M. Z., Cheng, R., Keutzer, K., ... & Goldberg, K. (2024). Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. *arXiv preprint arXiv:2409.03403*.

Tiger Task: Different Robot (UR5 + Robotiq)

- Source Policy: 90% on Source Robot
- 90% on Target Robot

Mirage (Cross-Painted)

Actual Target Robot

Source Domain Manipulation: Current Status

Simple Pick & Place tasks:

The success rate drop because of switching embodiments is under 10%.

More complex tasks:

Never tested.

For **UMI hardware, the localization precision is ~cm** so it cannot perform precise tasks (though the newest hardware PIKA has precision of ~mm).

On the other hand, image editing only makes visual features similar but **there are still geometry and dynamics differences**.

Embodiment-Agnostic Policy

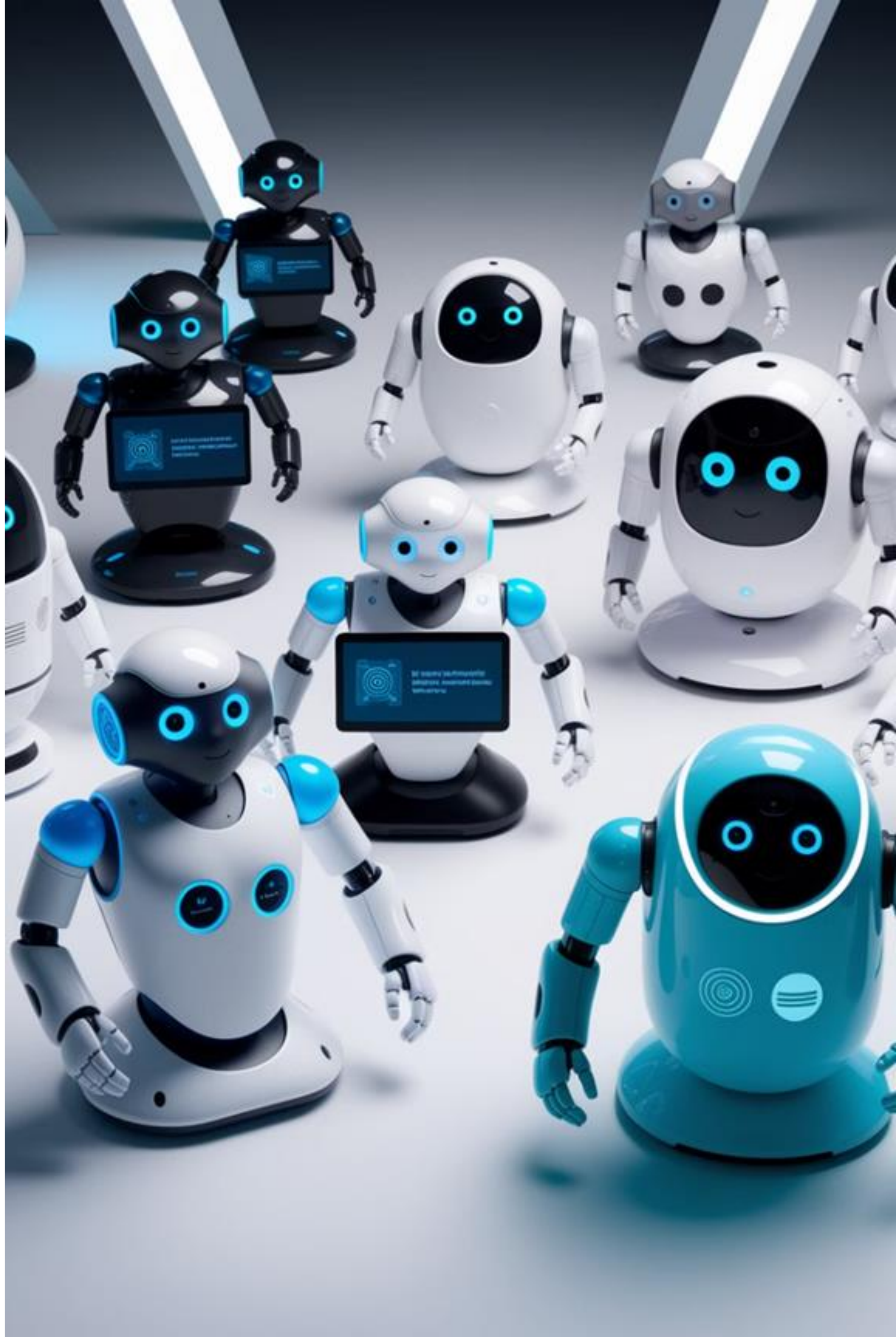
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- Source domain manipulation

2 Few-Shot Transfer Approaches

- **Domain-invariant transitions**
- Invariant features
- Hierarchical policies
- I don’t care, just train a large VLA



Use Domain-Invariant Transitions - Approach

Assuming **you don't have any real-world manipulation data of the target robot**, but **you have a physical simulator and an URDF model of the target robot**, which is a common case...

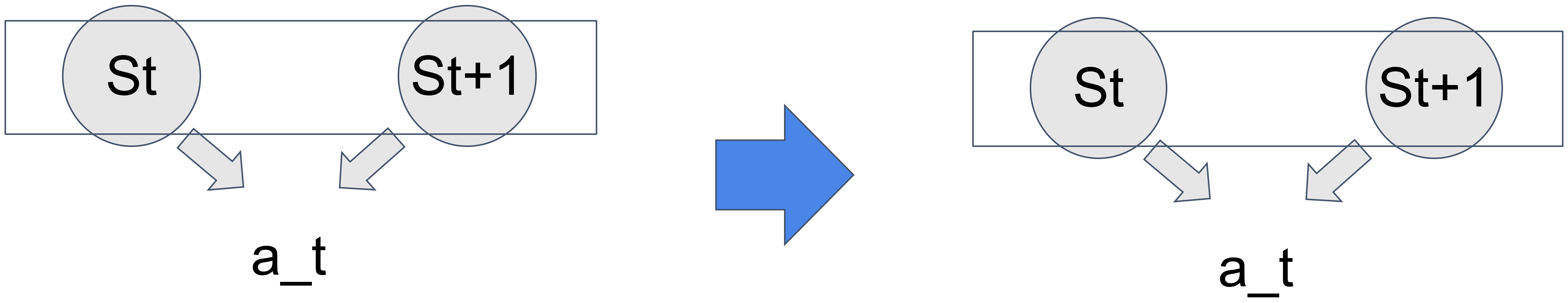
Use Domain-Invariant Transitions - MAIL

1. Train an inverse dynamic model (IDM) whose inputs are adjacent states of the manipulated object and outputs the intermediate action.
2. Use the IDM to infer action labels of target robot B, from manipulation videos of source robot A.

Salhotra, G., Liu, I., & Sukhatme, G. (2023). Learning robot manipulation from cross-morphology demonstration. *arXiv preprint arXiv:2304.03833*.

robot B state transitions

robot A state transitions



Train an IDM in simulation with URDF model of target robot B

Use IDM to infer actions of robot B from manipulation video of robot A

Embodiment-Agnostic Policy

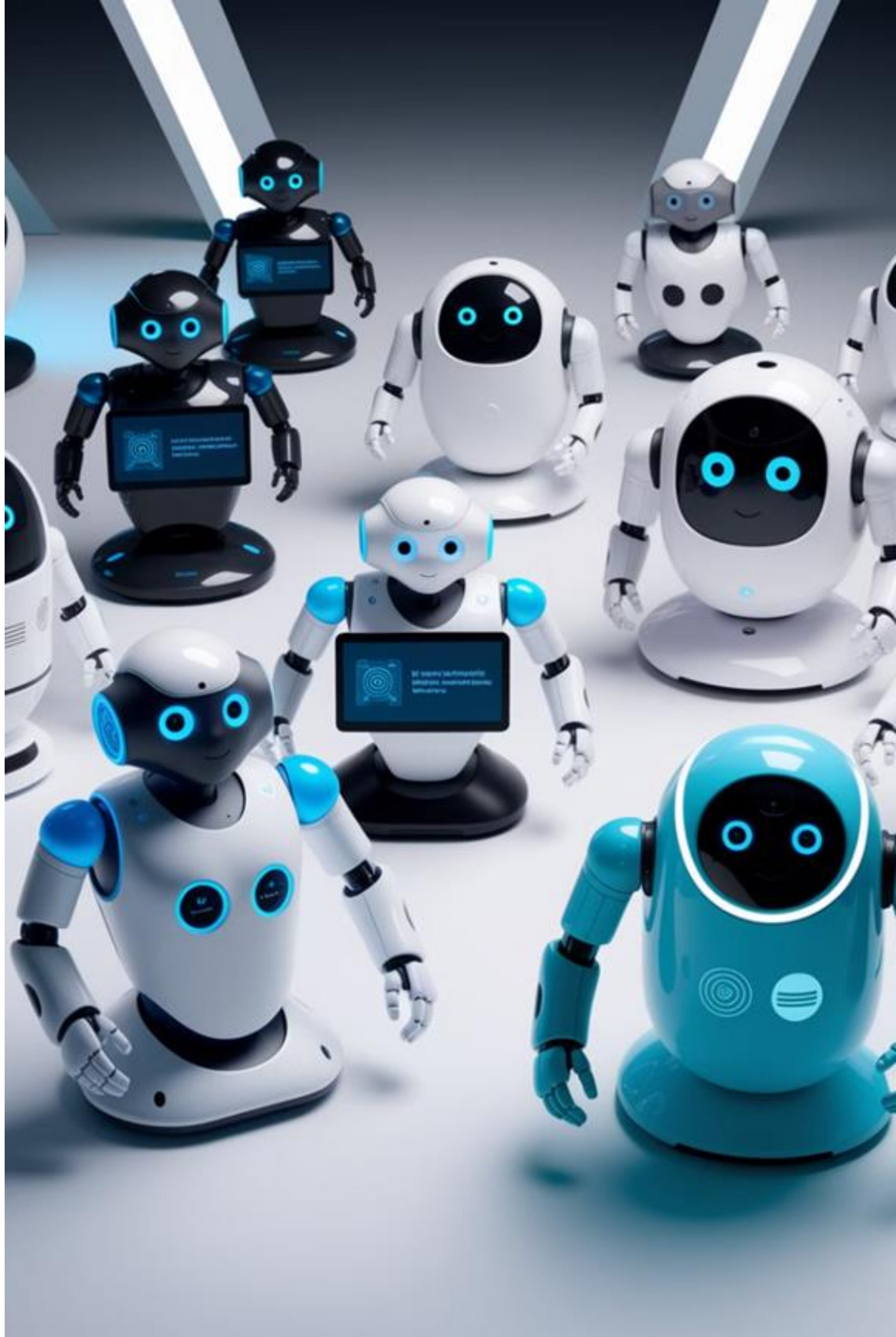
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- Source domain manipulation

2 Few-Shot Transfer Approaches

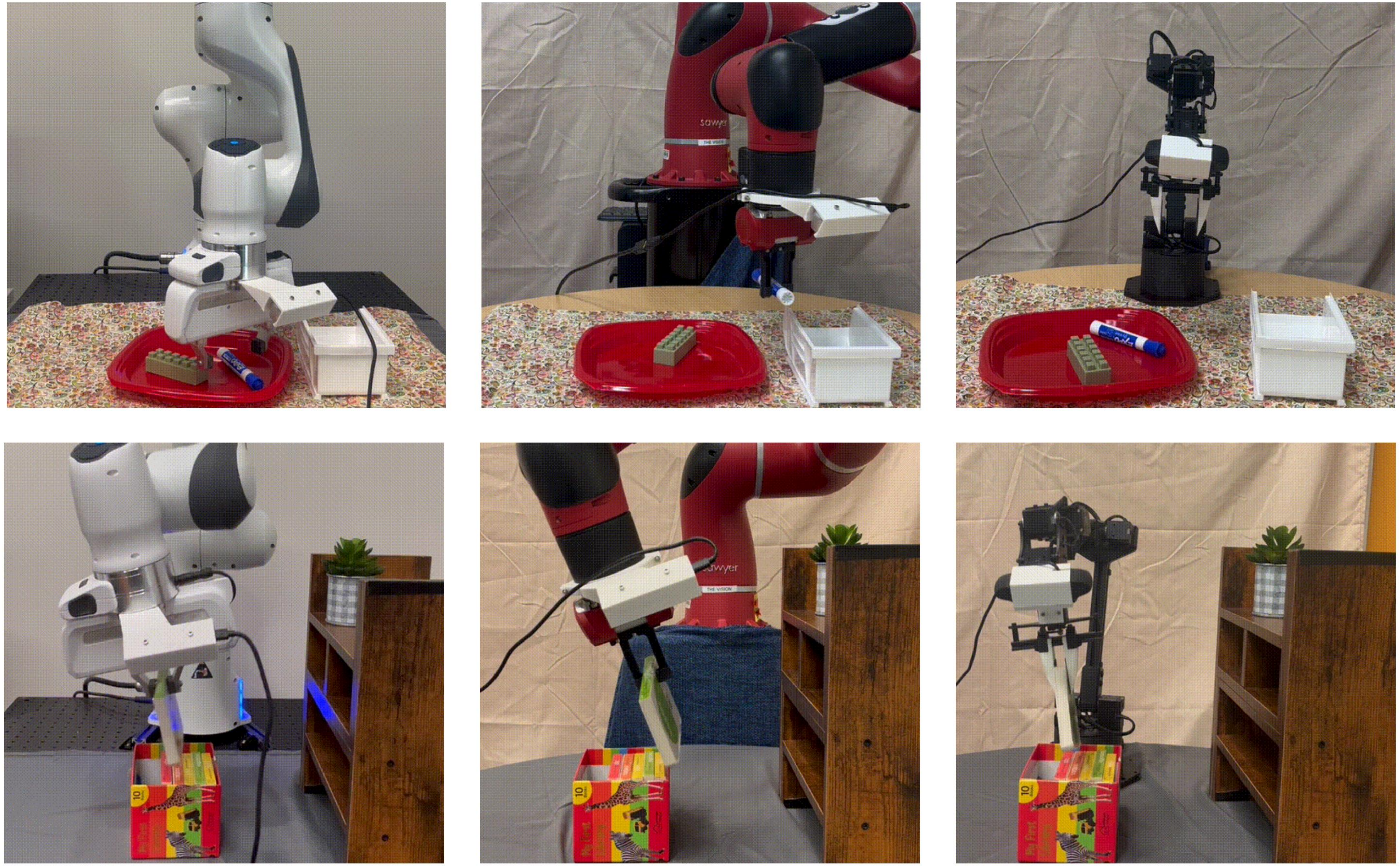
- Domain-invariant transitions
- **Invariant features**
- Hierarchical policies
- I don’t care, just train a large VLA



Invariant Features - Approaches

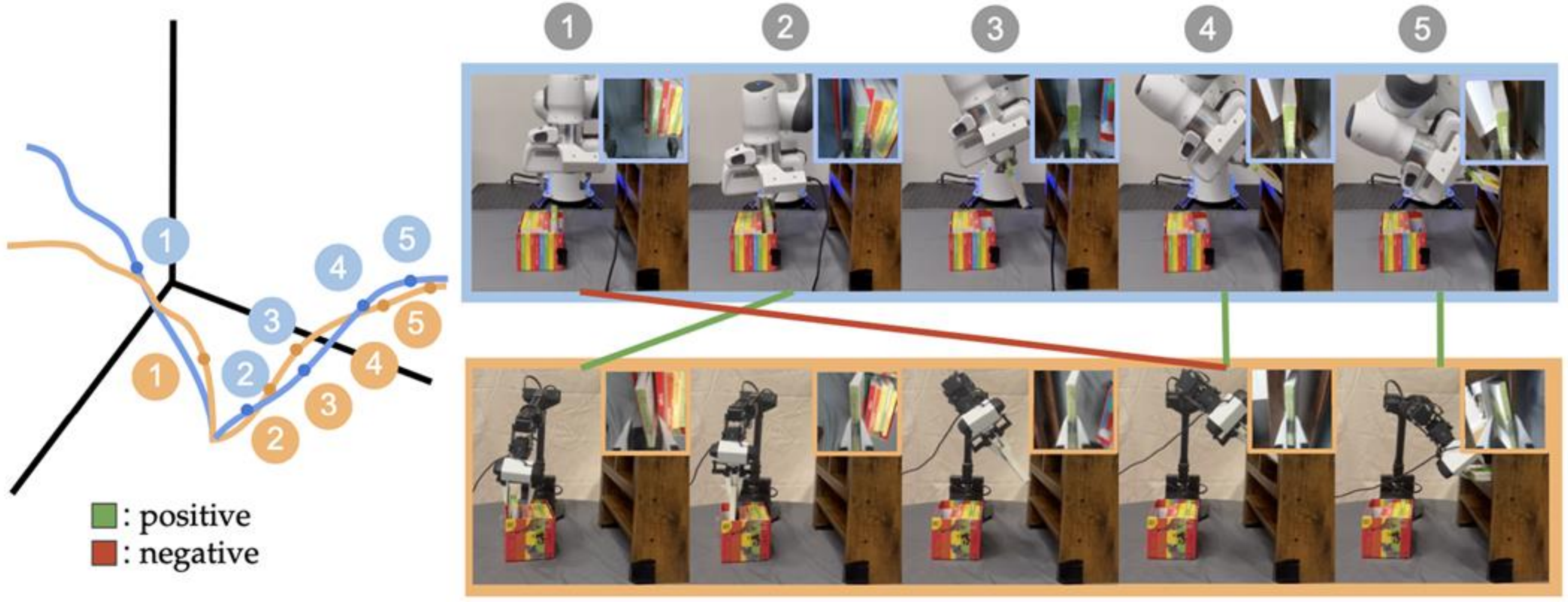
Do you have **paired data**? That means,, given a trajectory of source robot A and a trajectory of target robot B, **can you identify which states in the 2 trajectories are similar?**

Learn Invariant Features from Paired Data - Polybot



Learn Invariant Features from Paired Data - Polybot

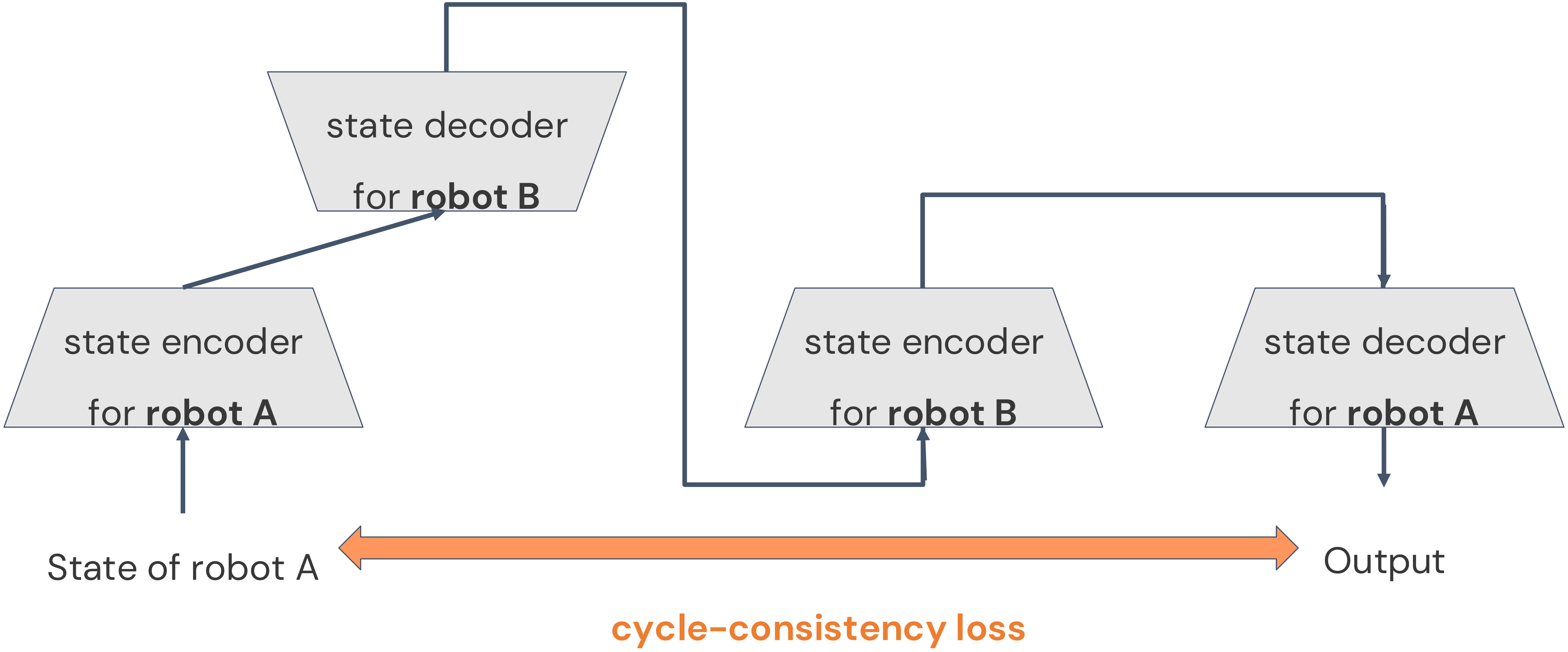
- Use contrastive learning to **bring the representations of similar states in the trajectories of different robotic arms closer together**
- Only use wrist camera views as visual input, i.e., no third-view cameras
- Use end effector poses as action representations



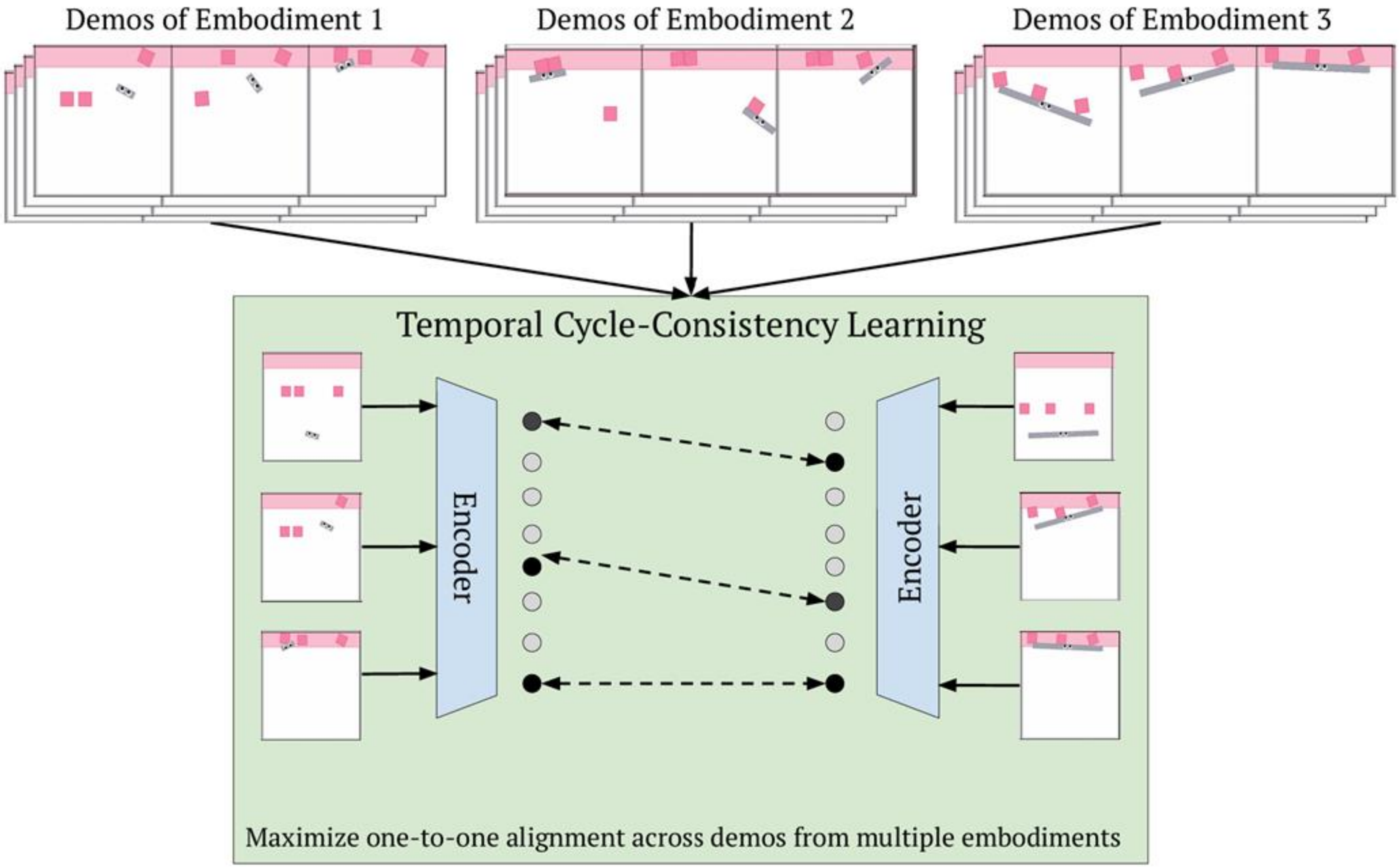
Yang, J., Sadigh, D., & Finn, C. (2023). Polybot: Training one policy across robots while embracing variability. *arXiv preprint arXiv:2307.03719*.

Learn Invariant Features from Unpaired Data

cycle-consistency loss, a simple example:



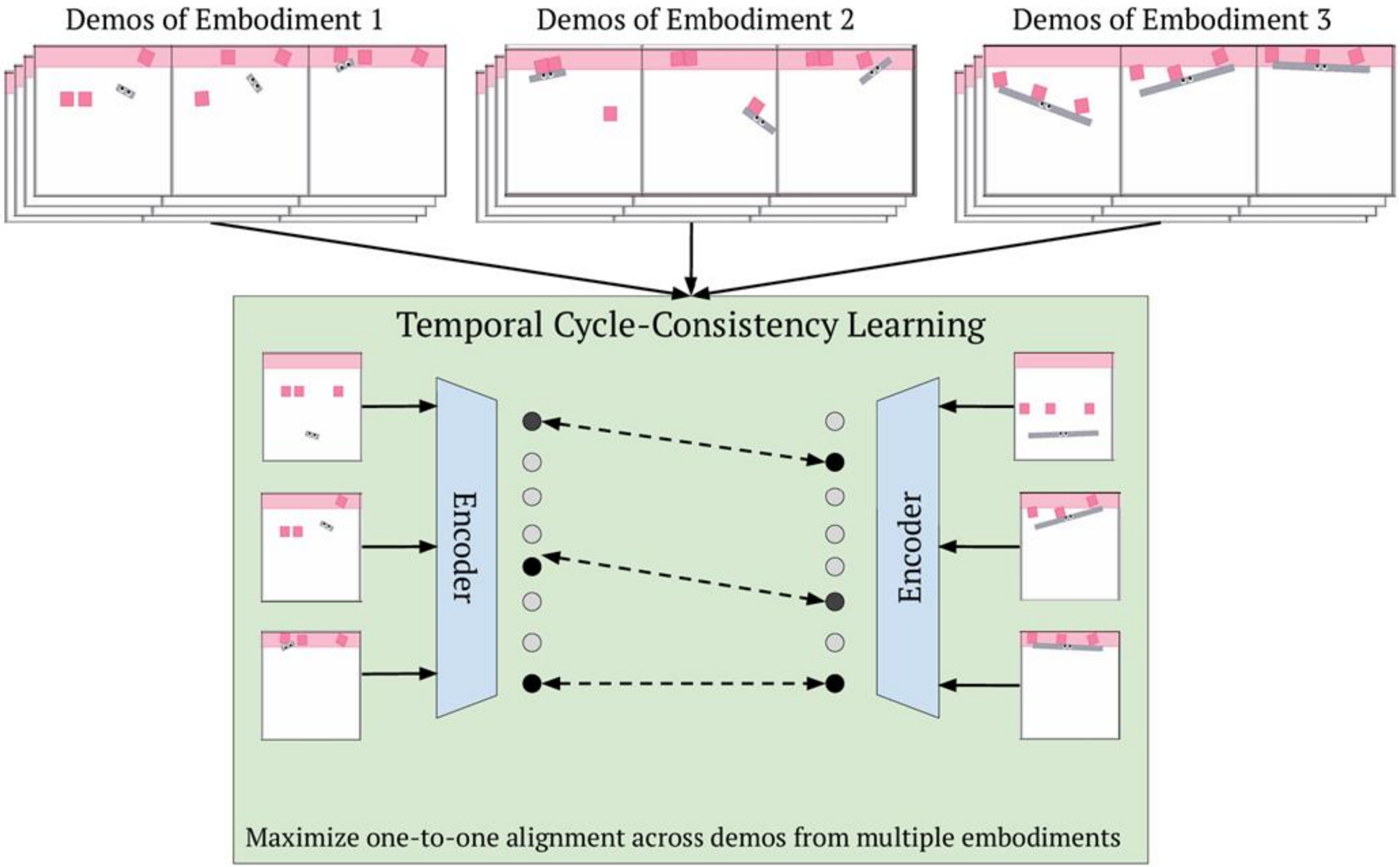
Learn Invariant Features from Unpaired Data - XIRL



First, the encoder is used to encode all video frames from all embodiments

Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., & Dwibedi, D. (2022, January). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning* (pp. 537-546). PMLR.

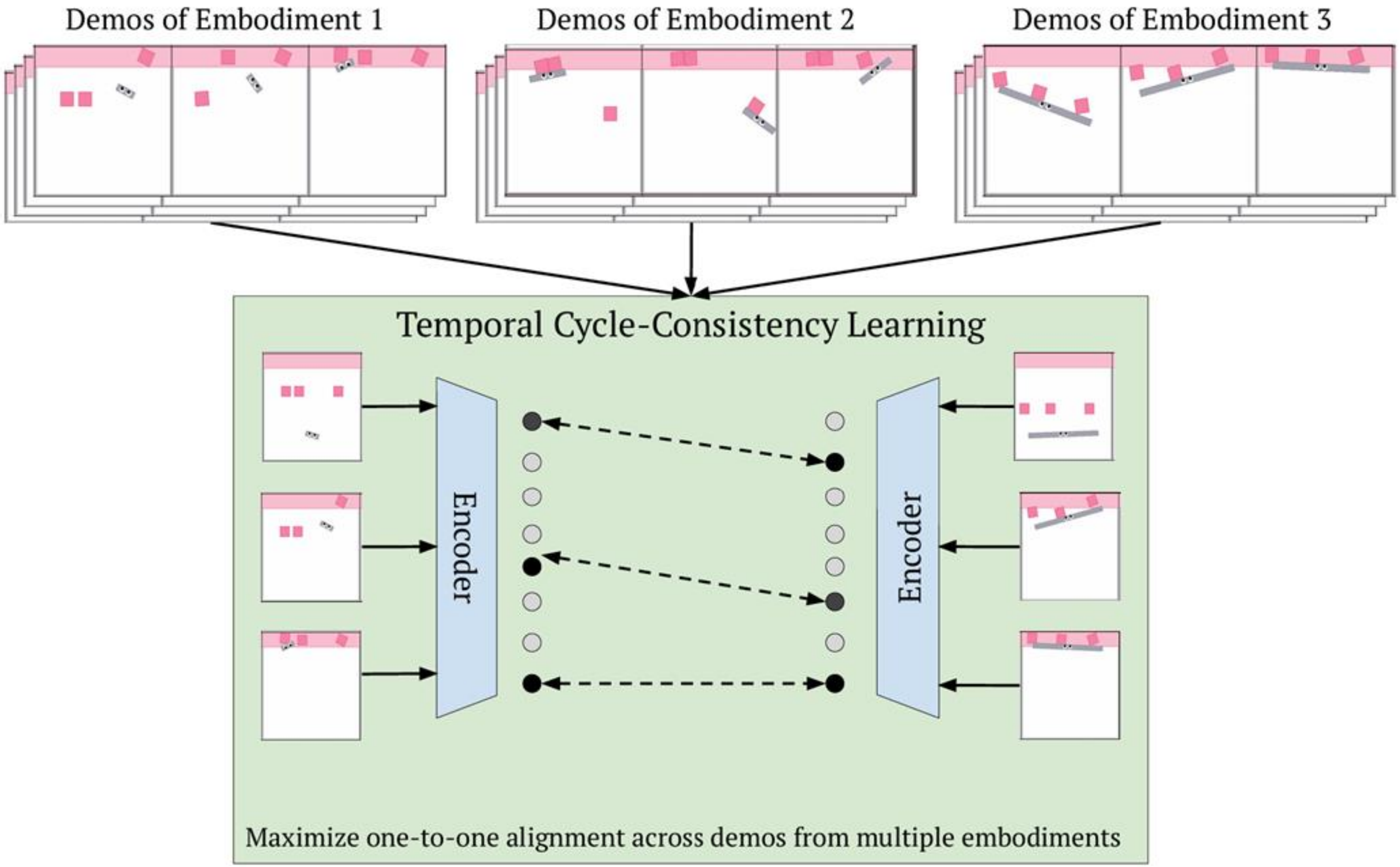
Learn Invariant Features from Unpaired Data - XIRL



Then, a frame encoding A1 is randomly selected from video A, and the similarity between A1 and the encodings of all frames in video B is calculated

Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., & Dwibedi, D. (2022, January). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning* (pp. 537-546). PMLR.

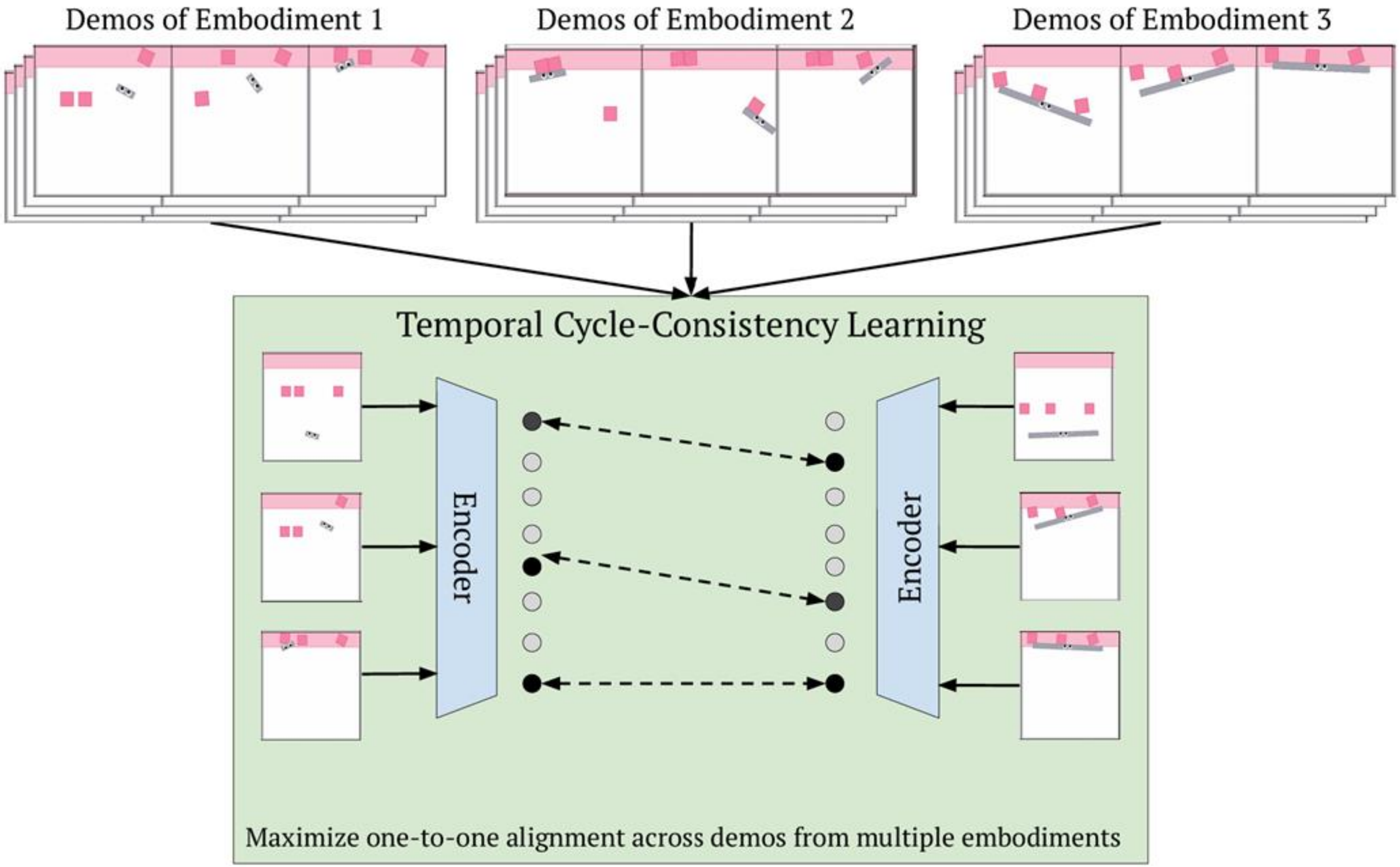
Learn Invariant Features from Unpaired Data - XIRL



Based on this similarity, a weighted combination of all frame encodings in video B is computed to obtain the soft nearest neighbor B1 of A1 in video B.

Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., & Dwibedi, D. (2022, January). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning* (pp. 537-546). PMLR.

Learn Invariant Features from Unpaired Data - XIRL



Next, the index of the frame in video A that is closest to B1 is identified. This index should be as close as possible to the index of A1.

Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., & Dwibedi, D. (2022, January). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning* (pp. 537-546). PMLR.

Learn Invariant Features from Unpaired Data - XIRL

XIRL proposed a **cycle-consistency loss** to train a video encoder.

- First, the encoder is used to encode all video frames from all embodiments
- Then, a frame encoding A1 is randomly selected from video A, and the similarity between A1 and the encodings of all frames in video B is calculated
- Based on this similarity, a weighted combination of all frame encodings in video B is computed to obtain the soft nearest neighbor B1 of A1 in video B.
- Next, the index of the frame in video A that is closest to B1 is identified. This index should be as close as possible to the index of A1.

Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., & Dwibedi, D. (2022, January). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning* (pp. 537-546). PMLR.

Learn Invariant Features - Challenges

How to scale up to N robots?

Embodiment-Agnostic Policy

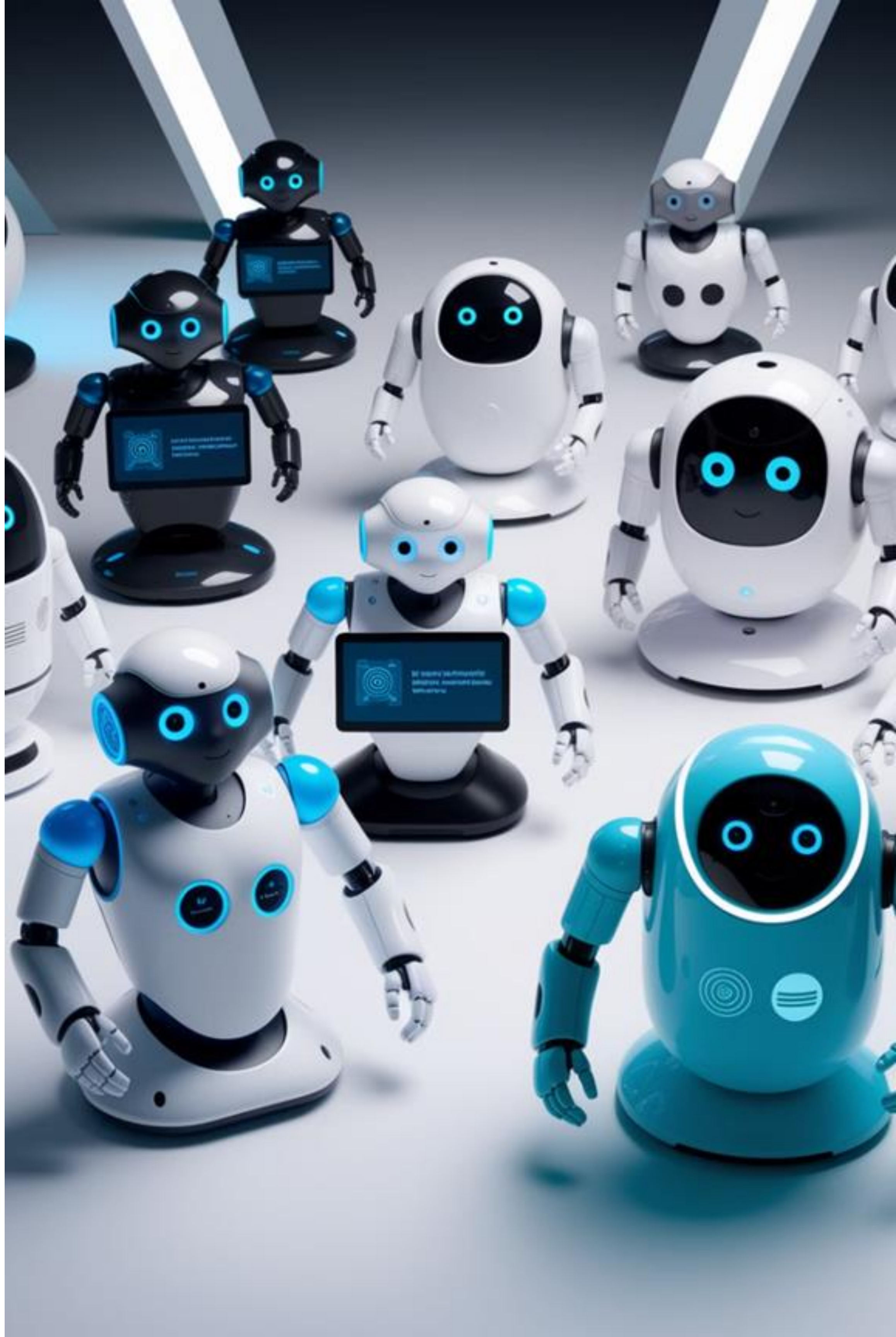
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- Source domain manipulation

2 Few-Shot Transfer Approaches

- Domain-invariant transitions
- Invariant features
- **Hierarchical policies**
- I don’t care, just train a large VLA



Hierarchical Policy

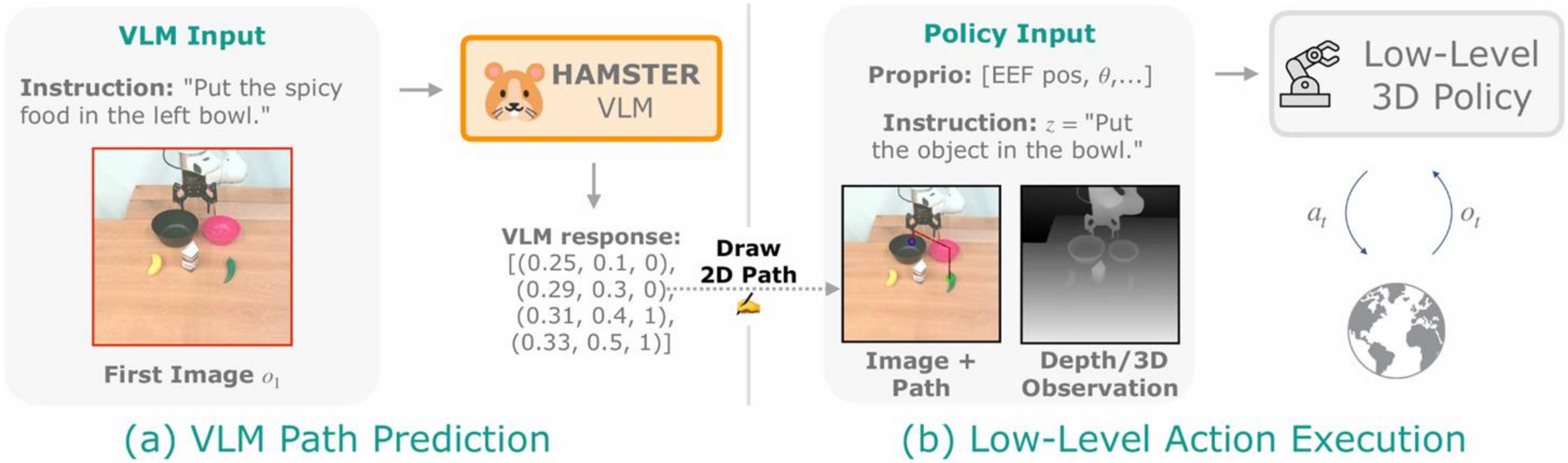
High-level policy outputs embodiment-agonistic actions

Low-level policy outputs embodiment-specific actions

HAMSTER

Hierarchical Action Models For Open-World Robot Manipulation

Hierarchical Policy - Hamster



Li, Y., Deng, Y., Zhang, J., Jang, J., Memme, M., Yu, R., ... & Goyal, A. (2025). HAMSTER: Hierarchical Action Models For Open-World Robot Manipulation. *arXiv preprint arXiv:2502.05485*.

Embodiment-Agnostic Policy

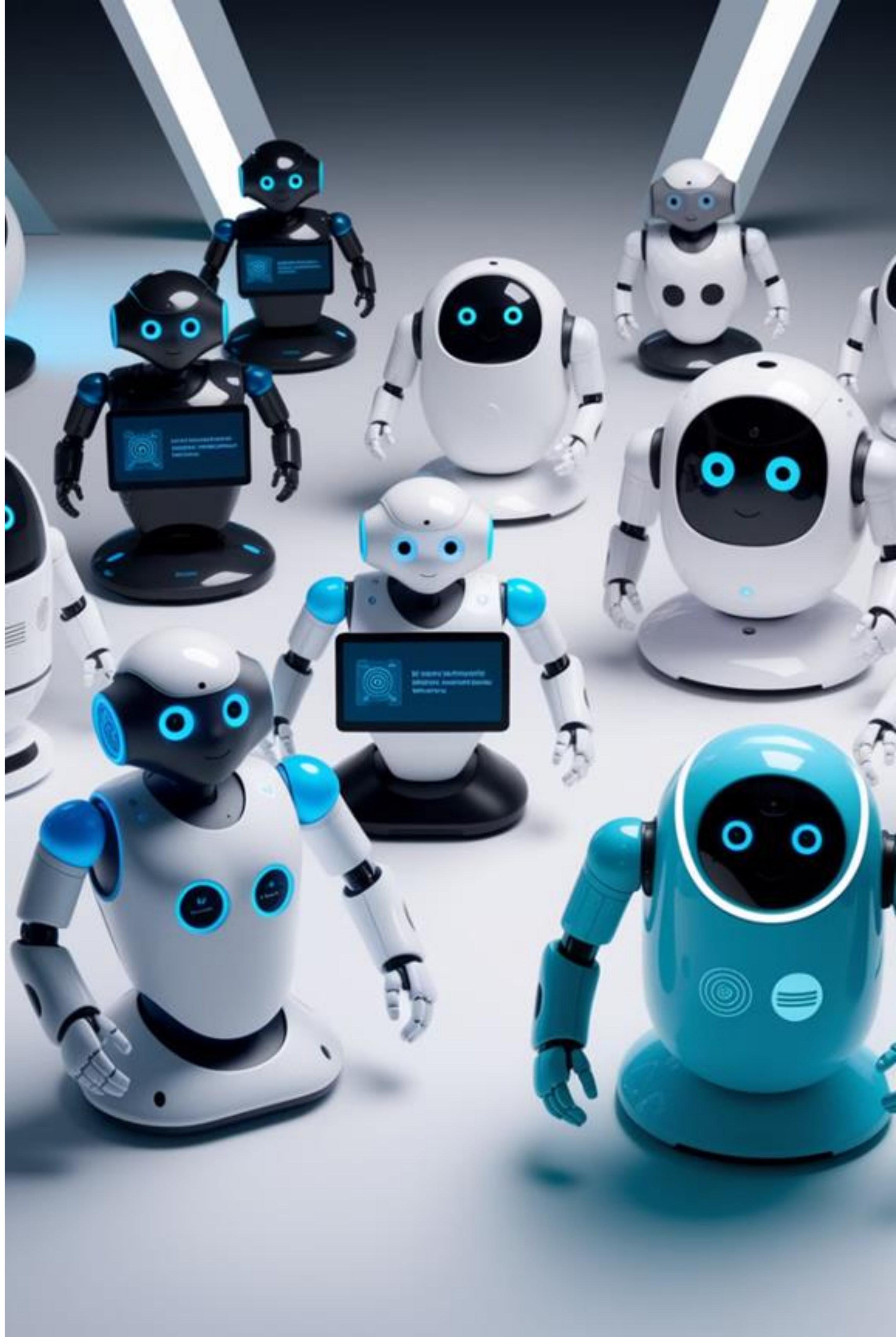
Make the input / output / internal representations similar for different robots

1 Zero-Shot Transfer Methods

- Source domain manipulation

2 Few-Shot Transfer Approaches

- Domain-invariant transitions
- Invariant features
- Hierarchical policies
- **I don’t care, just train a large VLA**



I don’t care, just train a large VLA

Action encoding

of RDT-1B

Index Range	Element Index	Mapped Physical Quantity
`[0, 10)`	0–9	Right arm joint positions
`[10, 15)`	10–14	Right gripper joint positions
`[15, 25)`	15–24	Right arm joint velocities
`[25, 30)`	25–29	Right gripper joint velocities
`[30, 33)`	30–32	Right end effector positions
`[33, 39)`	33–38	Right end effector 6D pose
`[39, 42)`	39–41	Right end effector velocities
`[42, 45)`	42–44	Right end effector angular velocities
`[45, 50)`	45–49	Reserved
`[50, 60)`	50–59	Left arm joint positions
`[60, 65)`	60–64	Left gripper joint positions
`[65, 75)`	65–74	Left arm joint velocities
`[75, 80)`	75–79	Left gripper joint velocities
`[80, 83)`	80–82	Left end effector positions
`[83, 89)`	83–88	Left end effector 6D pose
`[89, 92)`	89–91	Left end effector velocities
`[92, 95)`	92–94	Left end effector angular velocities
`[95, 100)`	95–99	Reserved
`[100, 102)`	100–101	Base linear velocities
`[102, 103)`	102	Base angular velocities
`[103, 128)`	103–127	Reserved

Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., ... & Zhu, J. (2024). Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*.

I don’t care, just train a large VLA

Does such VLA pretraining really helps downstream tasks?

I don't care, just train a large VLA

TABLE XIV: Ablation experiment: fine-tuning OpenVLA from scratch with the OFT recipe. Policy inputs here include a third-person camera image, a wrist camera image, robot proprioceptive state, and a language instruction. The from-scratch policies generally perform worse than the full OpenVLA-OFT policies, confirming that OpenVLA's pretrained representation is beneficial for downstream policy performance even when the fine-tuning recipe differs substantially from the pretraining recipe.

	Spatial SR (%)	Object SR (%)	Goal SR (%)	Long SR (%)	Average SR (%)
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
OpenVLA-OFT (scratch)	94.3	95.2	91.7	86.5	91.9

I don’t care, just train a large VLA

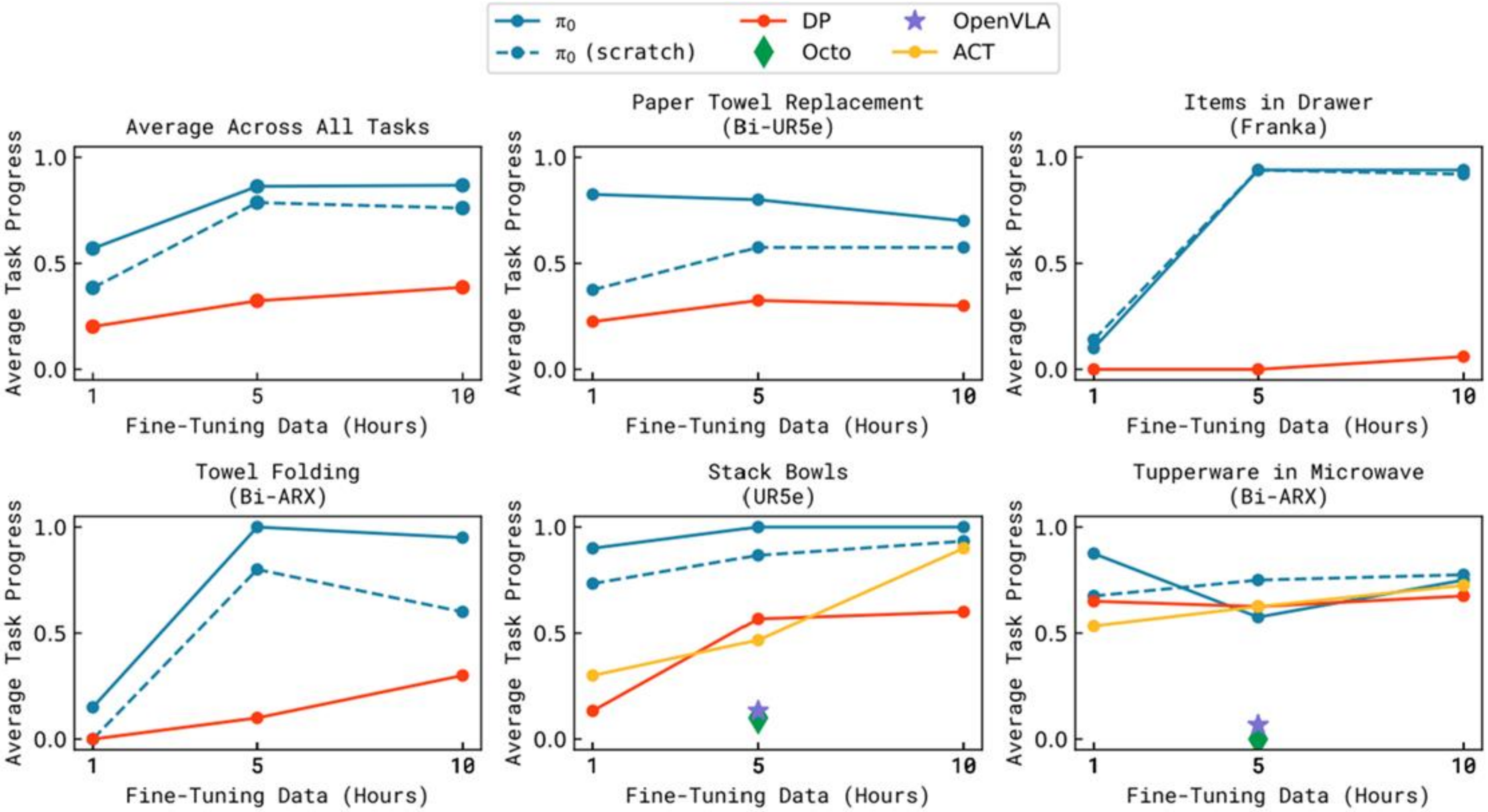


Fig. 11: **Fine-tuning with varying amounts of data.** π_0 can learn some easier tasks even with smaller amounts of data, and the pre-trained model often attains a larger improvement over the model trained from scratch.

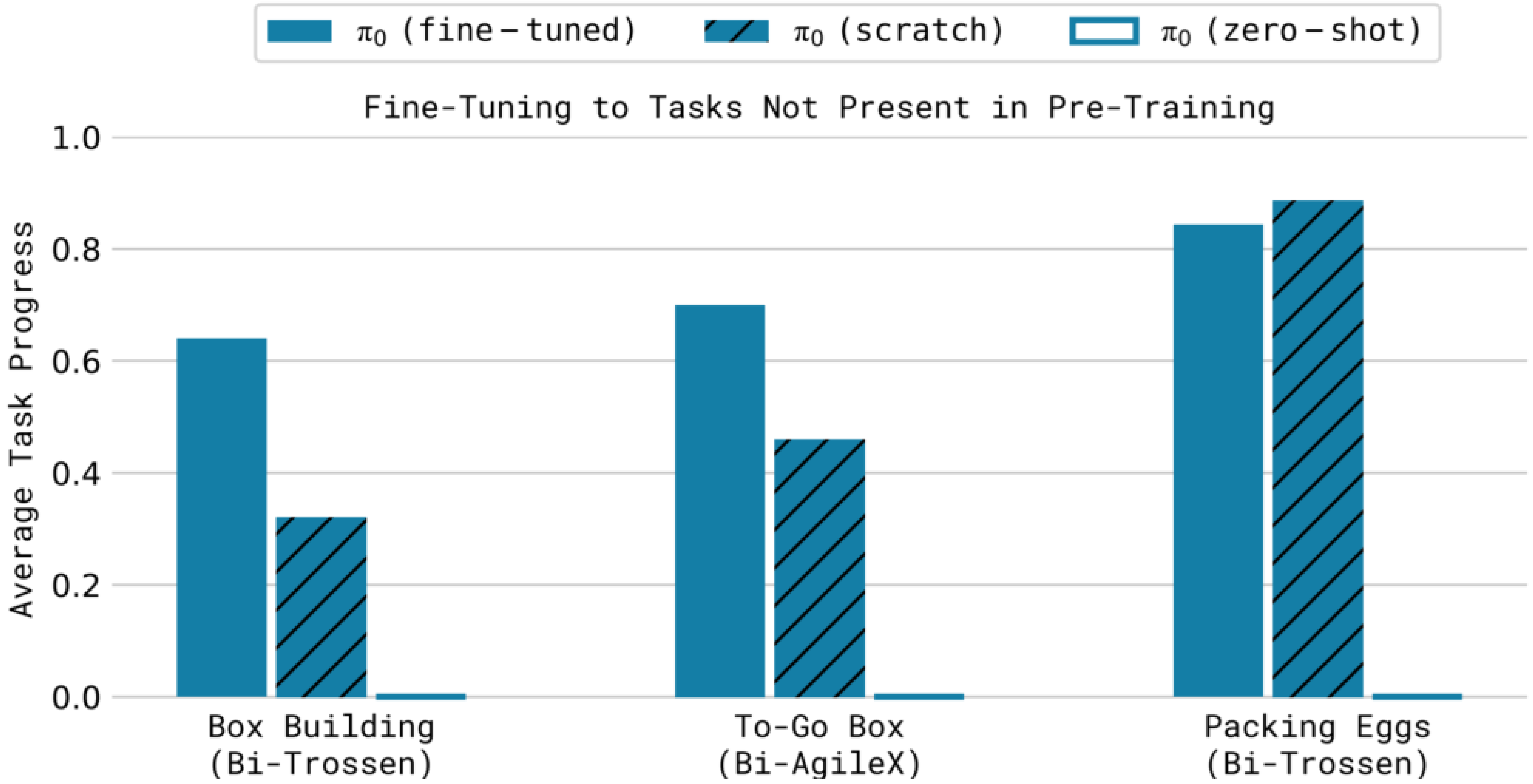
I don't care, just train a large VLA

If you have few finetuning data:

- Pretraining helps a lot

If you have >300 finetuning demos per task:

- Pretraining helps in some tasks, but not in others



I don’t care, just train a large VLA

If you have **few finetuning data**:

- Pretraining helps a lot

If you have **>300 finetuning demos per task**:

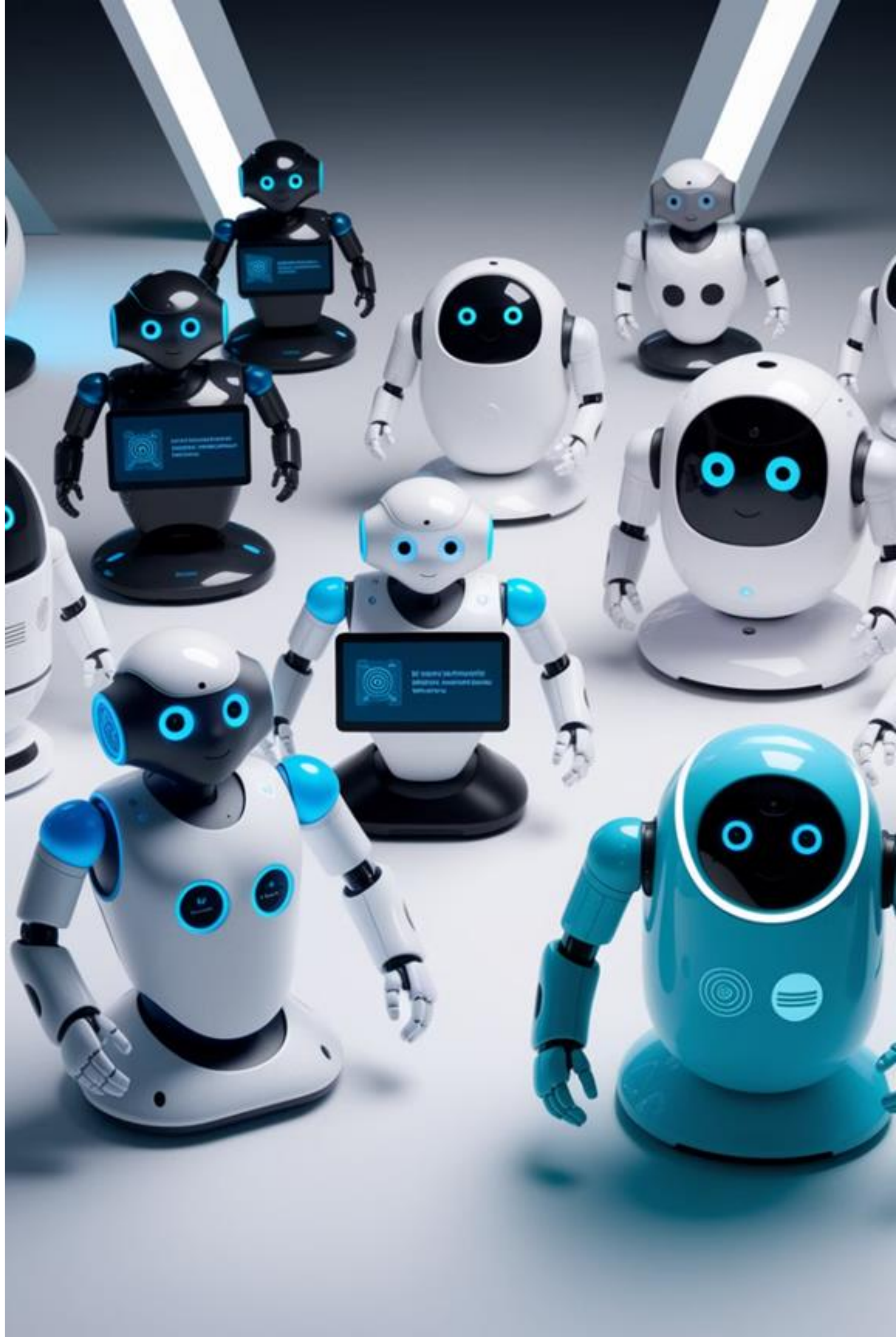
- Pretraining helps in some tasks, but not in others

Does the VLA learn **common knowledge from different robots**, instead of **learning independant implicit policies for different robots?**

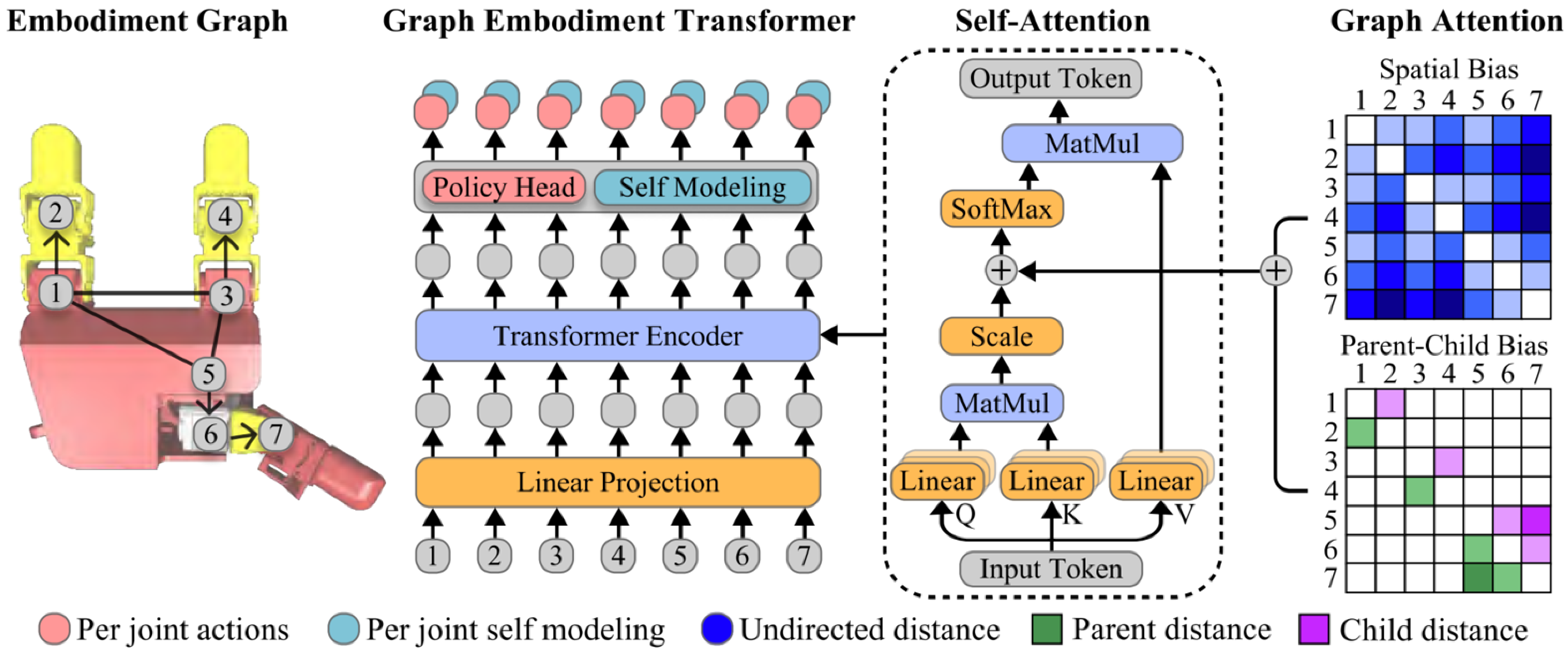
Embodiment-Aware Policy

Use the robotics arm geometric configuration / camera view perspectives as conditional input

1. **Geometric configuration as input**
2. Camera view perspectives as input (If you use 2D image input, rather than 3D point cloud input)



Geometric configuration as input



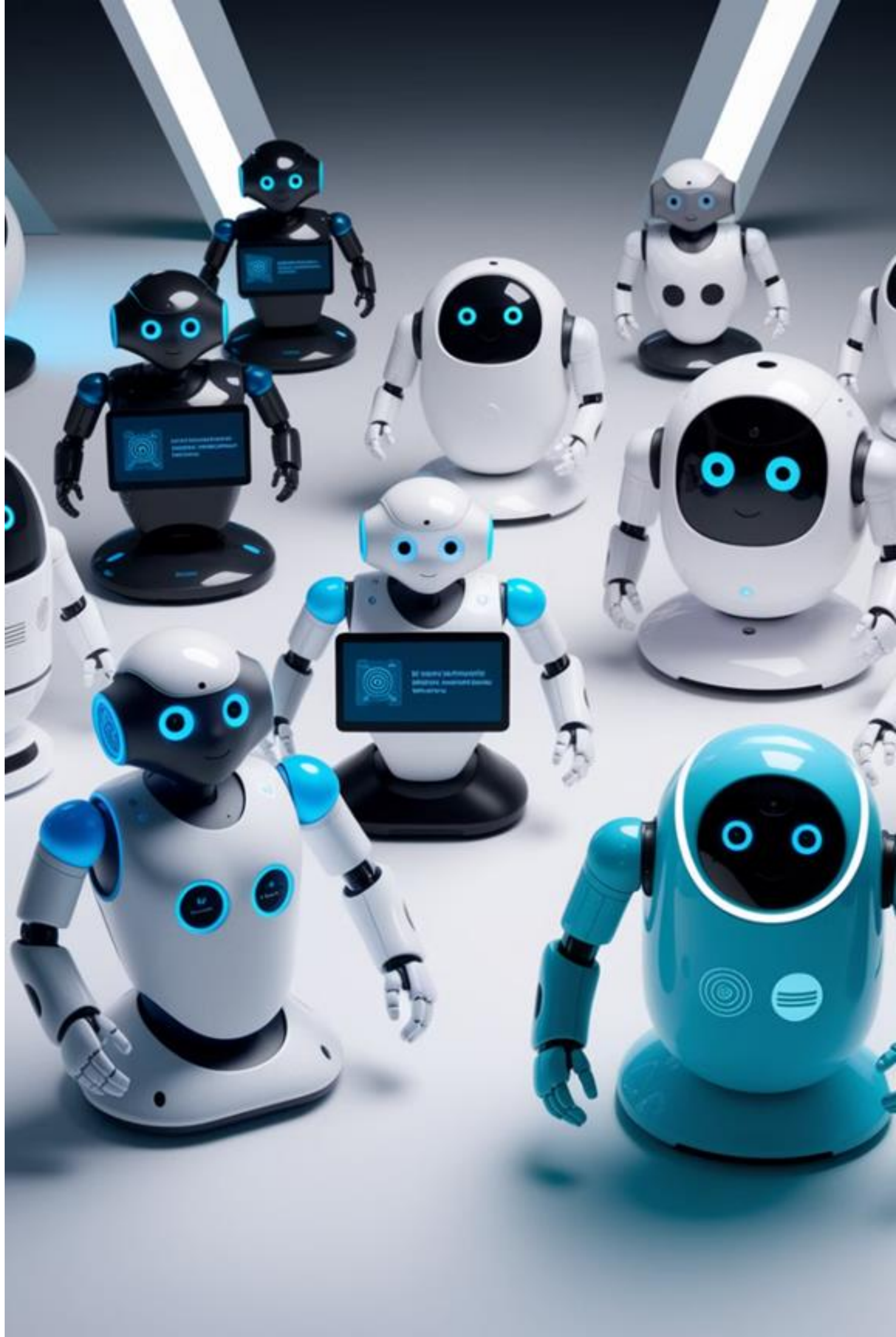
Patel, A., & Song, S. (2024). GET-Zero: Graph embodiment transformer for zero-shot embodiment generalization. *arXiv preprint arXiv:2407.15002*.

Embodiment-Aware Policy

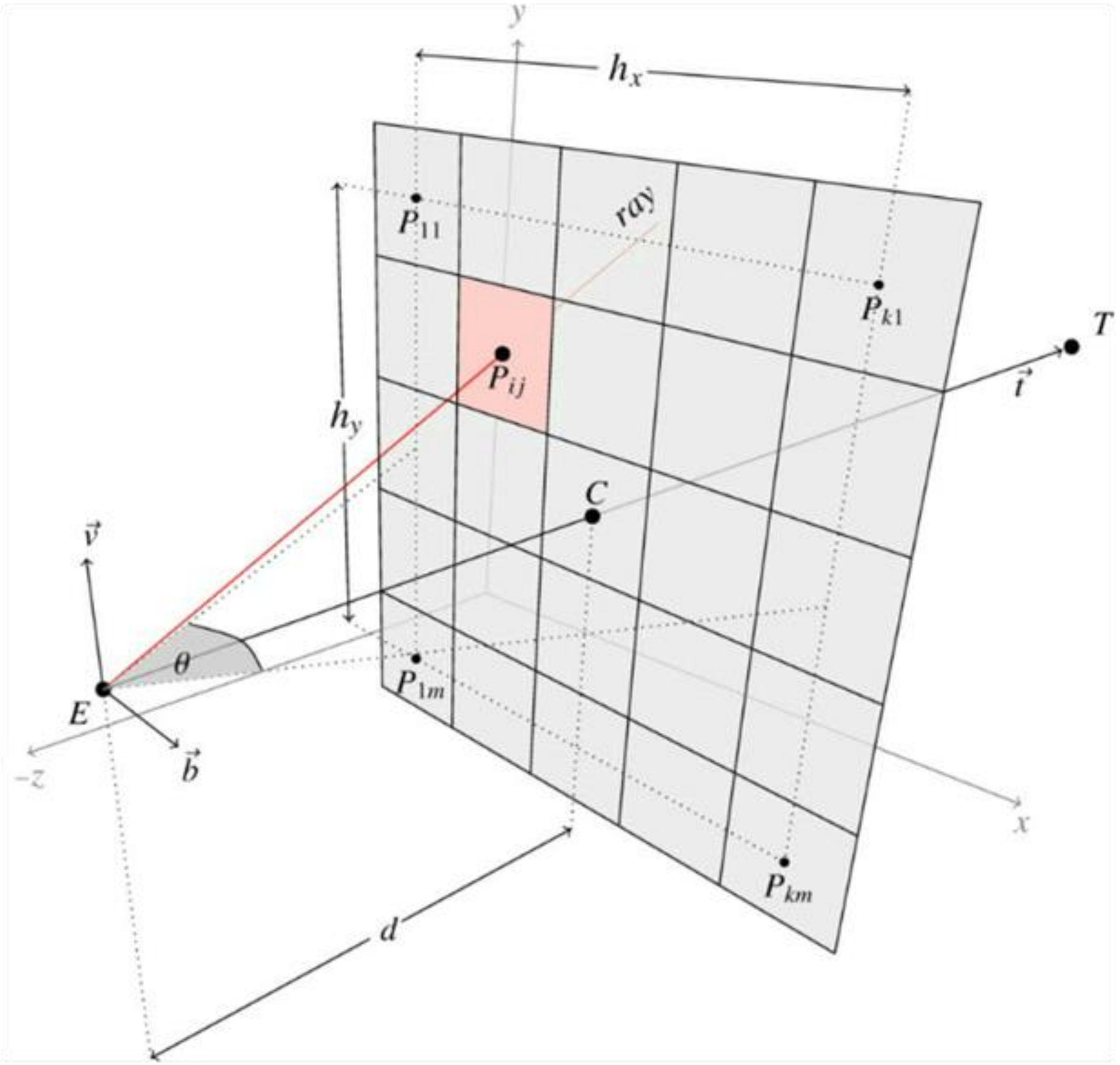
Use the robotics arm geometric configuration / camera view perspectives as conditional input

1 Geometric configuration as input

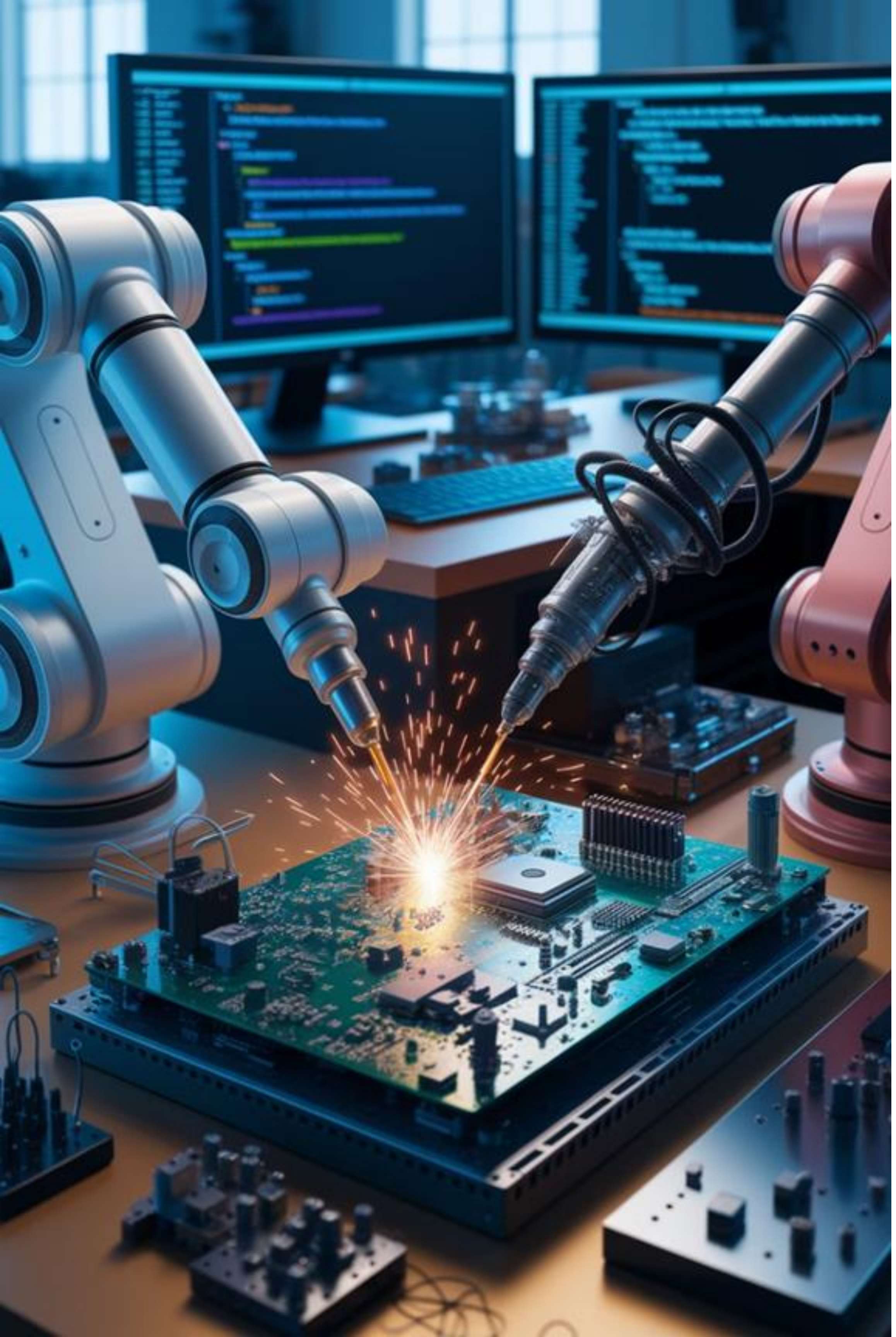
2 **Camera view perspectives as input**
(If you use 2D image input, rather than 3D point cloud input)



Camera view perspectives as input



Ray Direction Map:
the 3D unit vector from the camera to each pixel, with dimensions (channel, height, width).



Future

Short-term:
Use a hierarchical policy to reduce finetuning data of target robot

Long-term:
Pretrain an embodiment-aware policy