MolmoAct

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, Ranjay Krishna

Technical Report, Aug 2025

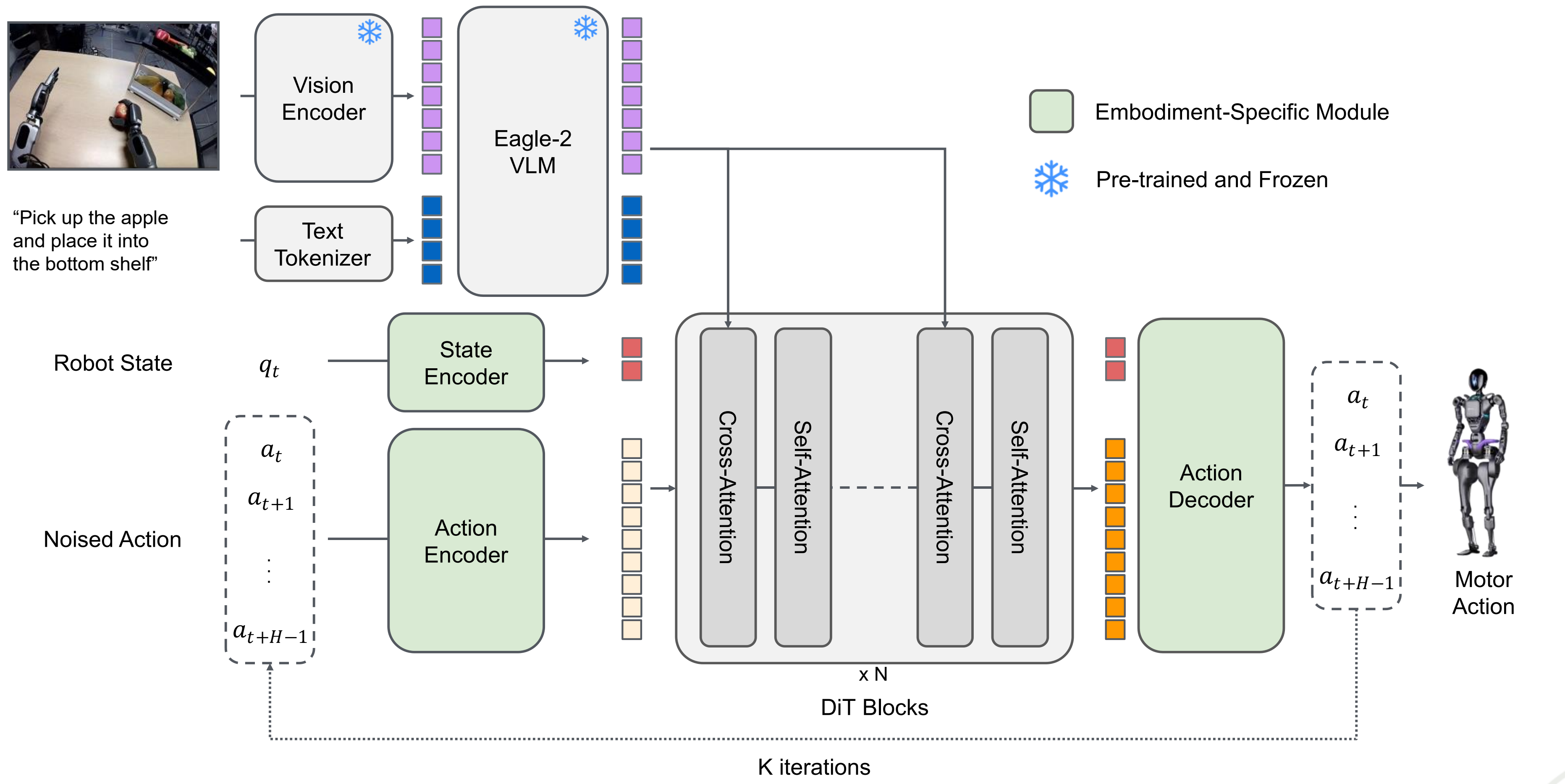
Presented by Shalin Jain and Siva Kailas

Georgia Tech

Outline

Vision Language Action Models so Far

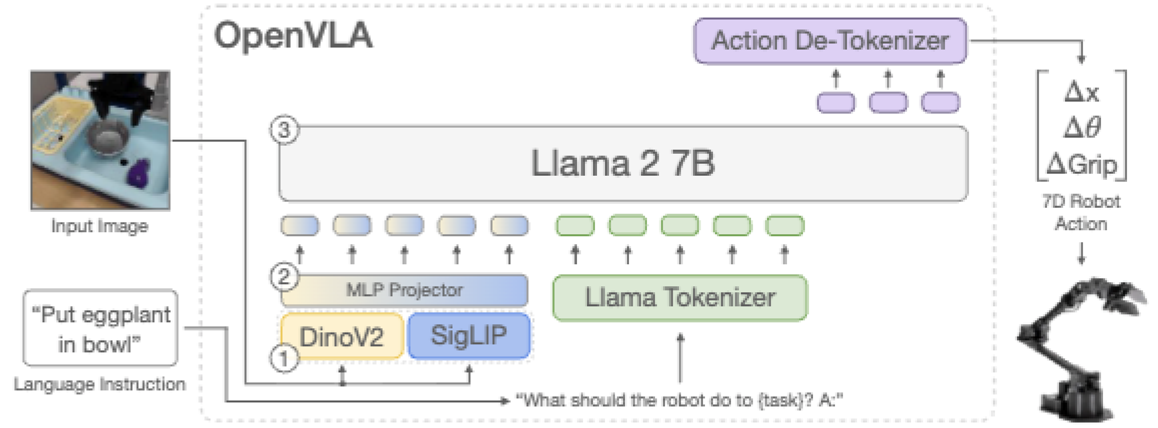
Directly Map Pixels and Language to Actions

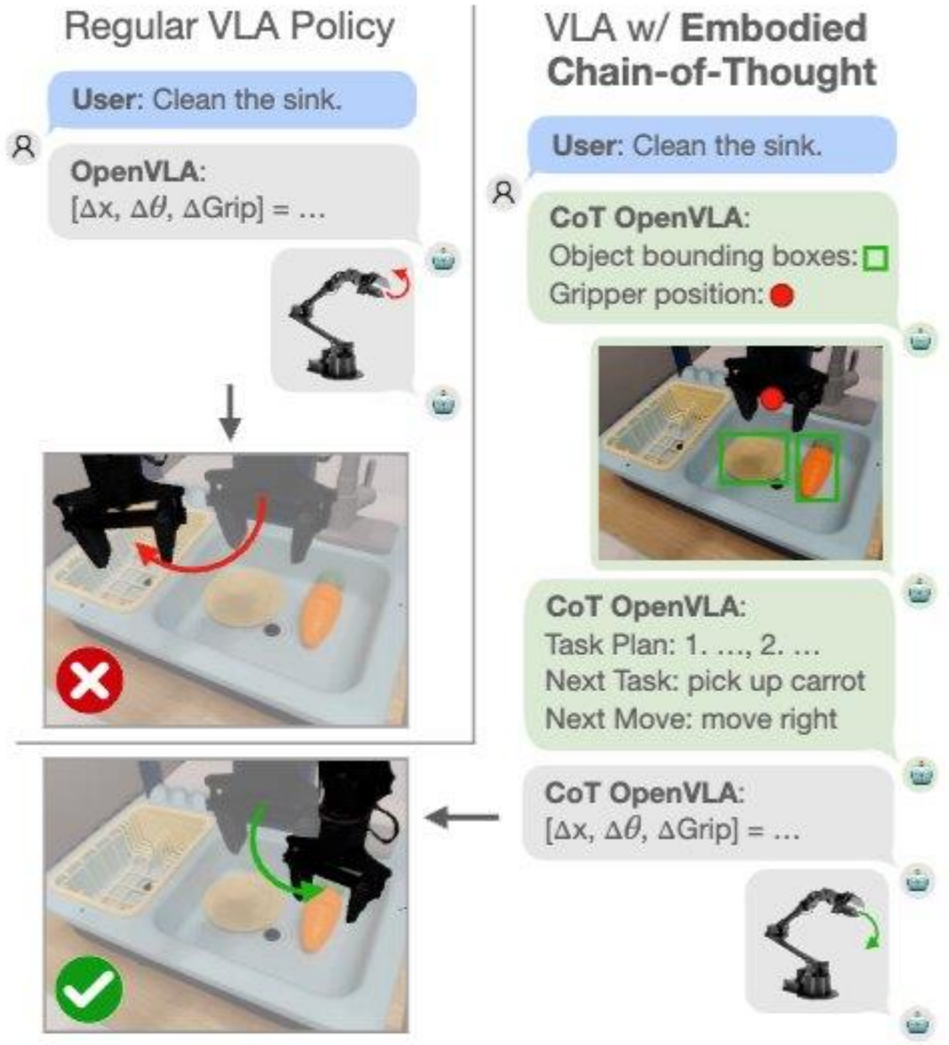


Example: gr00t n1 / n1.5 [1]

Vision Language Action Models so Far

Reason in Language and in the Image Space

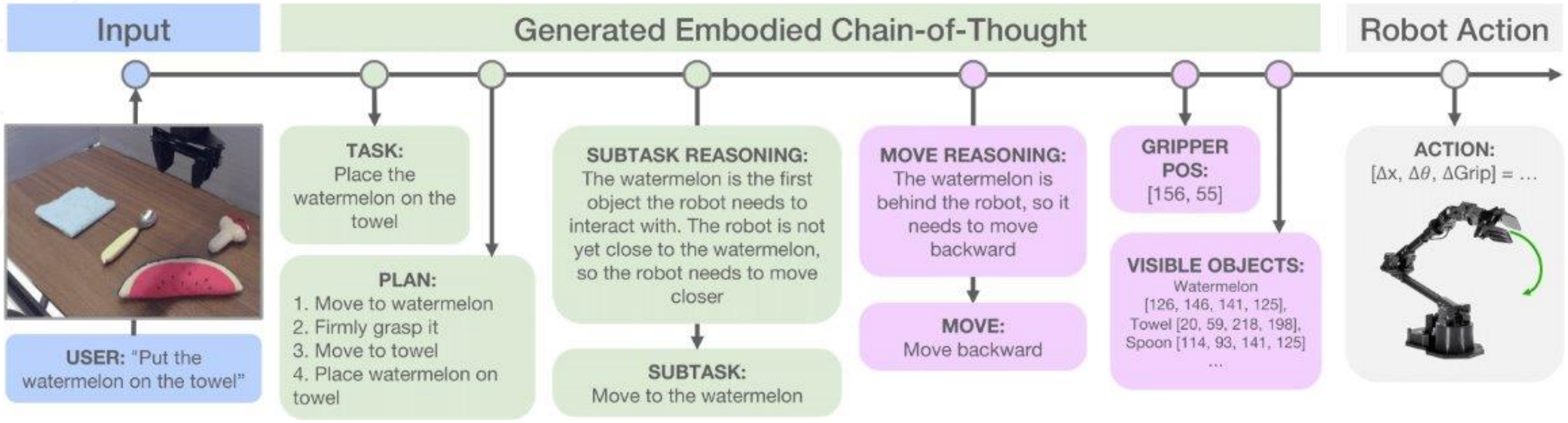




Regular VLA Policy
VLA w/ Embodied Chain-of-Thought
User: Clean the sink.
OpenVLA: [Δx, $\Delta\theta$, ΔGrip] = …
User: Clean the sink.
CoT OpenVLA: Object bounding boxes: 🟩 Gripper position: ●
CoT OpenVLA: Task Plan: 1. …, 2. … Next Task: pick up carrot Next Move: move right
CoT OpenVLA: [Δx, $\Delta\theta$, ΔGrip] = …

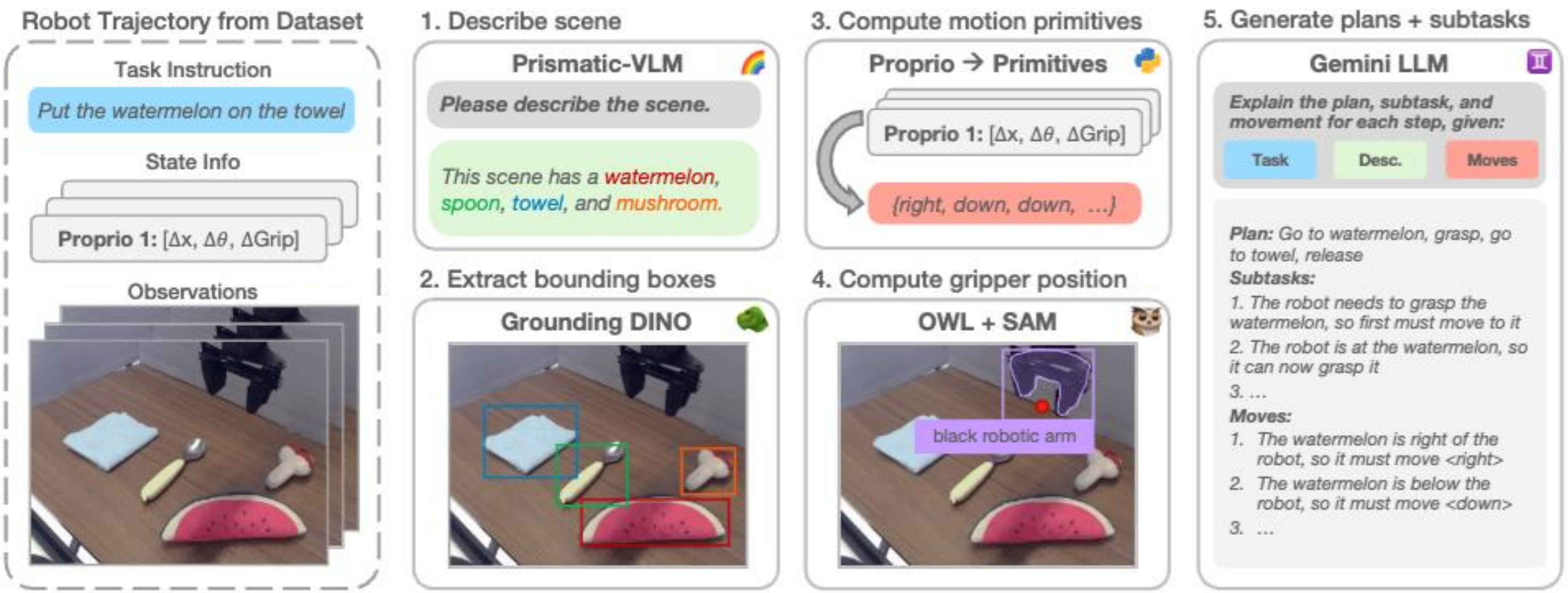
Example: Embodied Chain of Thought [2]

Embodied Chain of Thought - Method



Input
Generated Embodied Chain-of-Thought
Robot Action
USER: "Put the watermelon on the towel"
TASK: Place the watermelon on the towel
PLAN: 1. Move to watermelon 2. Firmly grasp it 3. Move to towel 4. Place watermelon on towel
SUBTASK REASONING: The watermelon is the first object the robot needs to interact with. The robot is not yet close to the watermelon, so the robot needs to move closer
SUBTASK: Move to the watermelon
MOVE REASONING: The watermelon is behind the robot, so it needs to move backward
MOVE: Move backward
GRIPPER POS: [156, 55]
VISIBLE OBJECTS: Watermelon [126, 146, 141, 125], Towel [20, 59, 218, 198], Spoon [114, 93, 141, 125] …
ACTION: [Δx, Δθ, ΔGrip] = …

Embodied Chain of Thought - Data



Robot Trajectory from Dataset
Task Instruction
Put the watermelon on the towel
State Info
Proprio 1: [Δx, Δθ, ΔGrip]
Observations
1. Describe scene
Prismatic-VLM
Please describe the scene.
This scene has a watermelon, spoon, towel, and mushroom.
2. Extract bounding boxes
Grounding DINO
3. Compute motion primitives
Proprio → Primitives
Proprio 1: [Δx, Δθ, ΔGrip]
{right, down, down, ...}
4. Compute gripper position
OWL + SAM
black robotic arm
5. Generate plans + subtasks
Gemini LLM
Explain the plan, subtask, and movement for each step, given:
Task
Desc.
Moves

Plan: Go to watermelon, grasp, go to towel, release

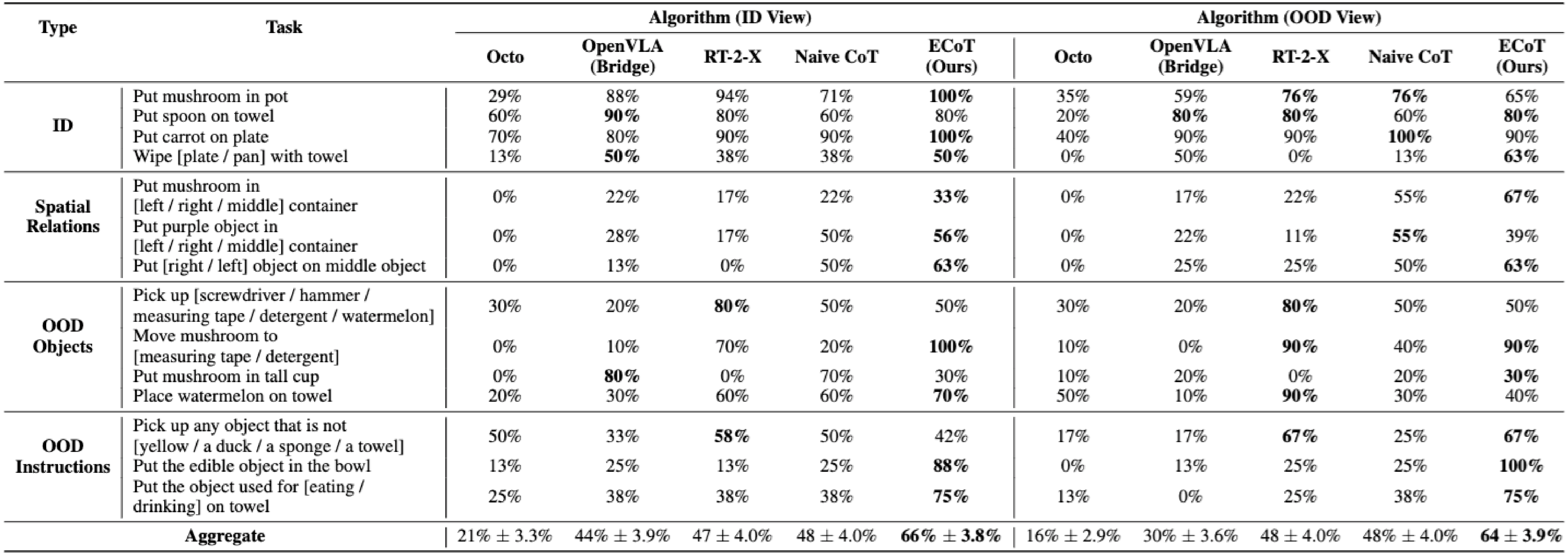
Subtasks:

1. The robot needs to grasp the watermelon, so first must move to it
2. The robot is at the watermelon, so it can now grasp it
3. ...

Moves:

1. The watermelon is right of the robot, so it must move <right>
2. The watermelon is below the robot, so it must move <down>
3. ...

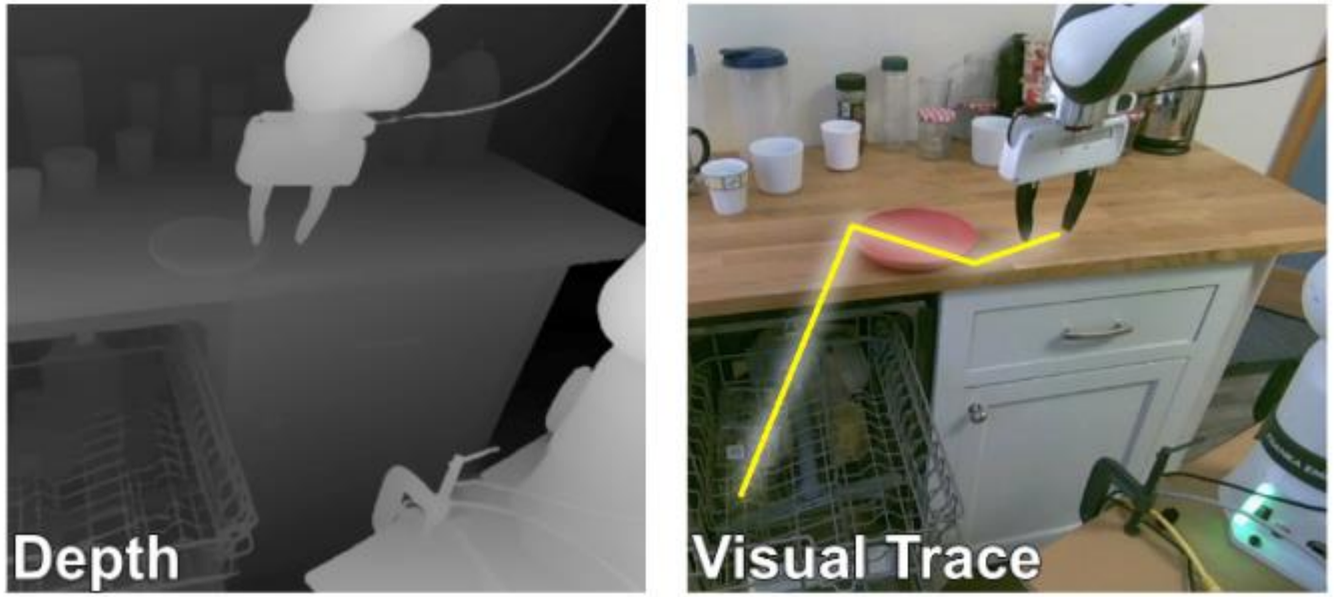
Embodied Chain of Thought - Results



Type	Task	Algorithm (ID View)					Algorithm (OOD View)				
		Octo	OpenVLA (Bridge)	RT-2-X	Naive CoT	ECoT (Ours)	Octo	OpenVLA (Bridge)	RT-2-X	Naive CoT	ECoT (Ours)
ID	Put mushroom in pot	29%	88%	94%	71%	**100%**	35%	59%	**76%**	**76%**	65%
	Put spoon on towel	60%	**90%**	80%	60%	80%	20%	**80%**	**80%**	60%	**80%**
	Put carrot on plate	70%	80%	90%	90%	**100%**	40%	90%	90%	**100%**	90%
	Wipe [plate / pan] with towel	13%	**50%**	38%	38%	**50%**	0%	50%	0%	13%	**63%**
Spatial Relations	Put mushroom in [left / right / middle] container	0%	22%	17%	22%	**33%**	0%	17%	22%	55%	**67%**
	Put purple object in [left / right / middle] container	0%	28%	17%	50%	**56%**	0%	22%	11%	**55%**	39%
	Put [right / left] object on middle object	0%	13%	0%	50%	**63%**	0%	25%	25%	50%	**63%**
OOD Objects	Pick up [screwdriver / hammer / measuring tape / detergent / watermelon]	30%	20%	**80%**	50%	50%	30%	20%	**80%**	50%	50%
	Move mushroom to [measuring tape / detergent]	0%	10%	70%	20%	**100%**	10%	0%	**90%**	40%	**90%**
	Put mushroom in tall cup	0%	**80%**	0%	70%	30%	10%	20%	0%	20%	**30%**
	Place watermelon on towel	20%	30%	60%	60%	**70%**	50%	10%	**90%**	30%	40%
OOD Instructions	Pick up any object that is not [yellow / a duck / a sponge / a towel]	50%	33%	**58%**	50%	42%	17%	17%	**67%**	25%	**67%**
	Put the edible object in the bowl	13%	25%	13%	25%	**88%**	0%	13%	25%	25%	**100%**
	Put the object used for [eating / drinking] on towel	25%	38%	38%	38%	**75%**	13%	0%	25%	38%	**75%**
Aggregate		21% ± 3.3%	44% ± 3.9%	47 ± 4.0%	48 ± 4.0%	**66% ± 3.8%**	16% ± 2.9%	30% ± 3.6%	48 ± 4.0%	48% ± 4.0%	**64 ± 3.9%**

MolmoAct Contributions

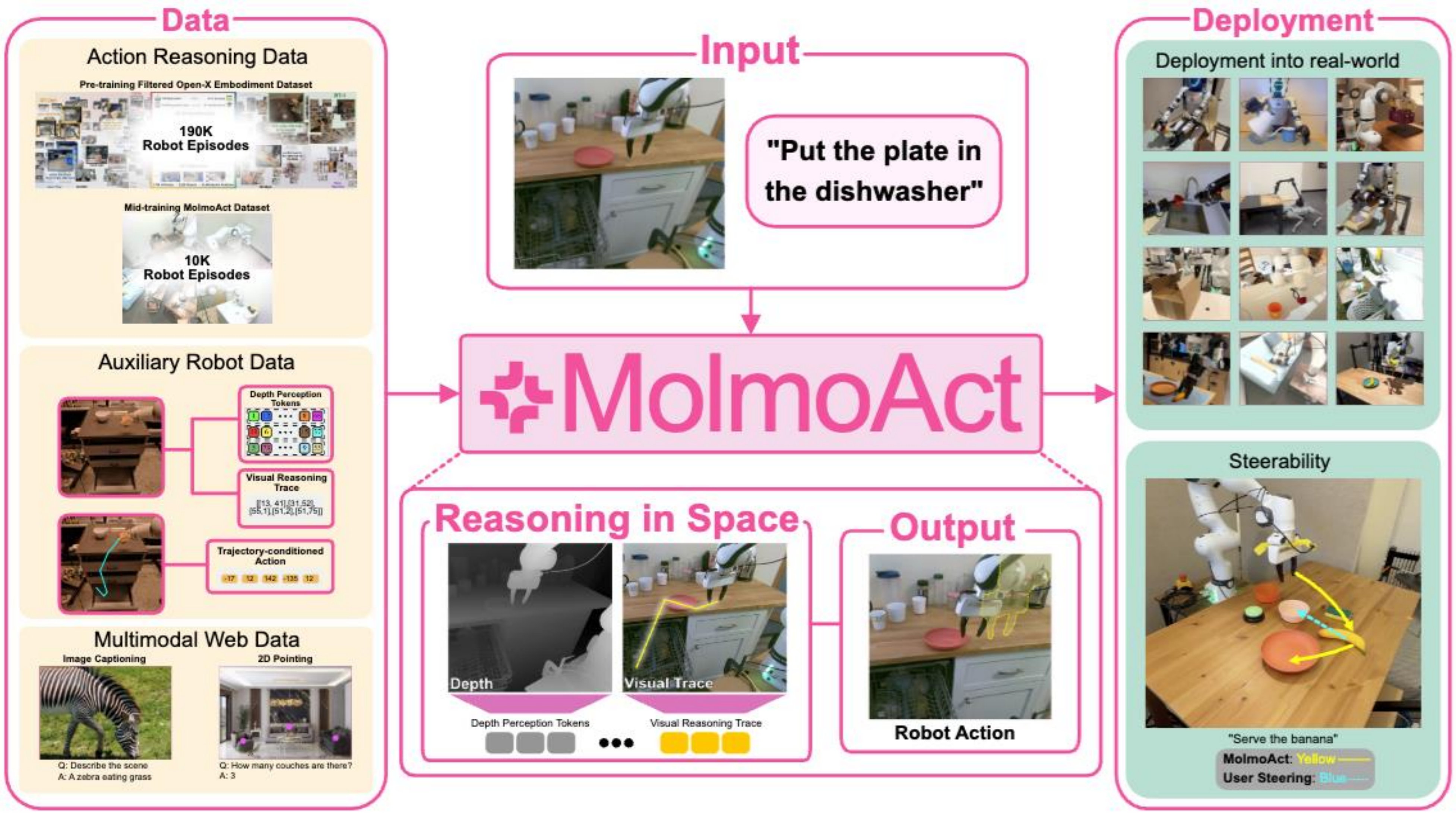
1. Reasoning in 3d space
2. Reasoning about trajectory planning
3. Recipe for converting existing robot vision action datasets to support reasoning



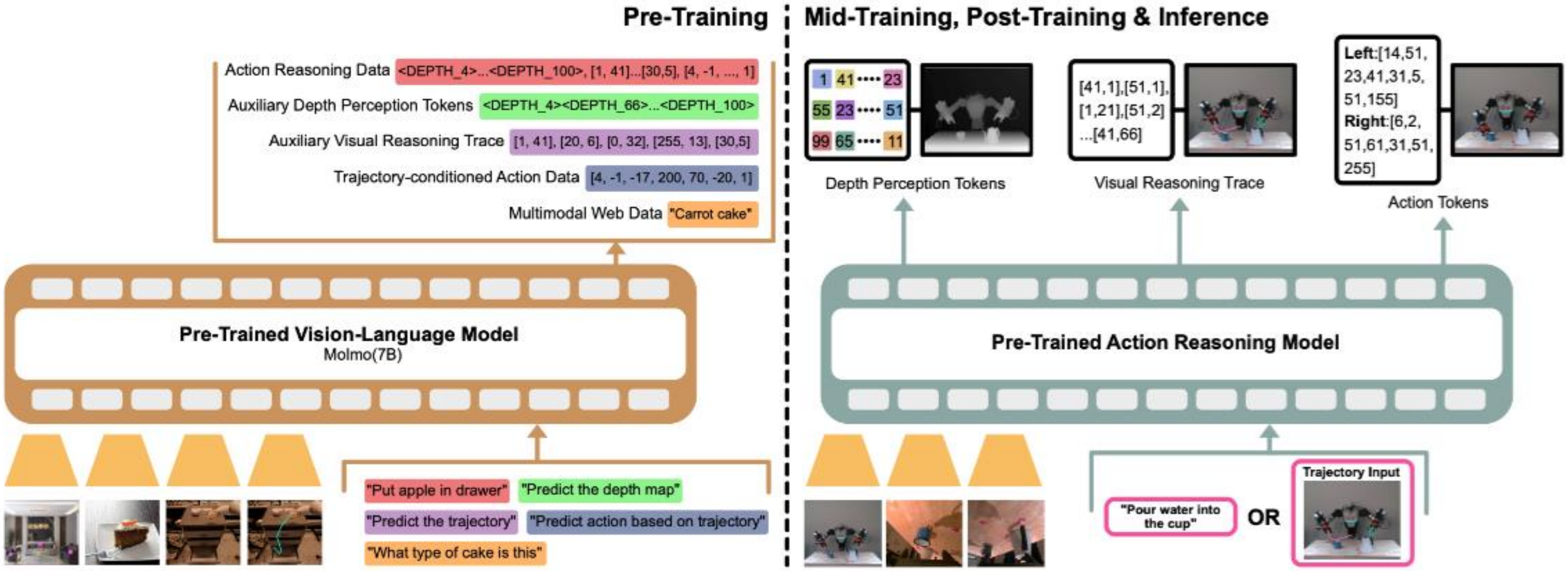
[3] MolmoAct

Towards Action Reasoning Models

MolmoAct Overview

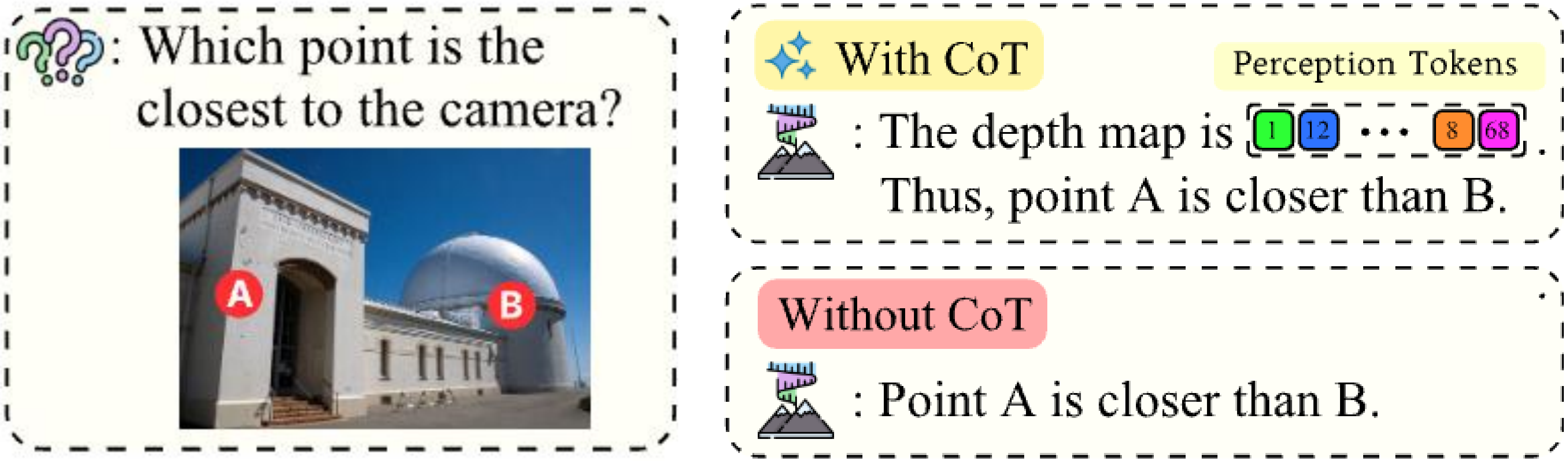


Model Overview


Pre-Training
Mid-Training, Post-Training & Inference
Action Reasoning Data <DEPTH_4>...<DEPTH_100>, [1, 41]...[30,5], [4, -1, ..., 1]
Auxiliary Depth Perception Tokens <DEPTH_4><DEPTH_66>...<DEPTH_100>
Auxiliary Visual Reasoning Trace [1, 41], [20, 6], [0, 32], [255, 13], [30,5]
Trajectory-conditioned Action Data [4, -1, -17, 200, 70, -20, 1]
Multimodal Web Data "Carrot cake"
Pre-Trained Vision-Language Model
Molmo(7B)
"Put apple in drawer"
"Predict the depth map"
"Predict the trajectory"
"Predict action based on trajectory"
"What type of cake is this"
1 41 23 55 23 51 99 65 11
Depth Perception Tokens
[41,1],[51,1],[1,21],[51,2]...[41,66]
Visual Reasoning Trace
Left:[14,51,23,41,31,5,51,155] Right:[6,2,51,61,31,51,255]
Action Tokens
Pre-Trained Action Reasoning Model
"Pour water into the cup"
OR
Trajectory Input

Depth Perception Tokens

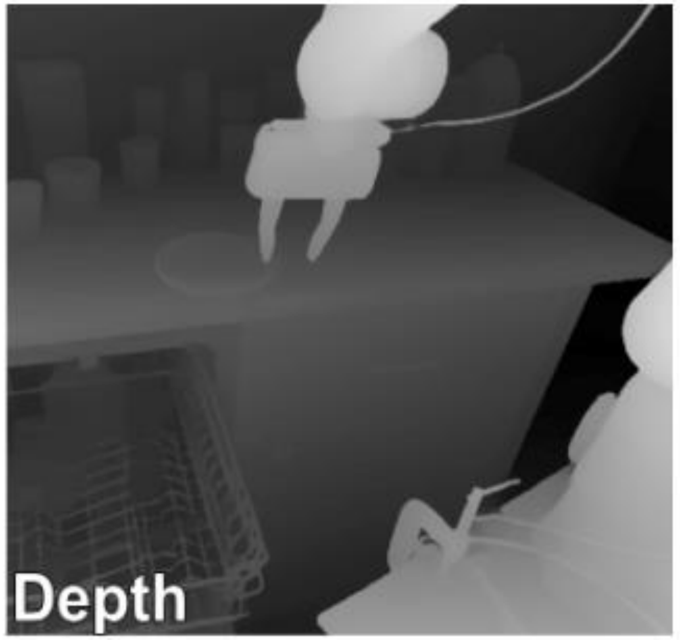
Why Depth?


: Which point is the closest to the camera?
A
B
With CoT
Perception Tokens
: The depth map is 1 12 ··· 8 68 .
Thus, point A is closer than B.
Without CoT
: Point A is closer than B.

[4] LLaVA-AURORA

Depth Perception Tokens

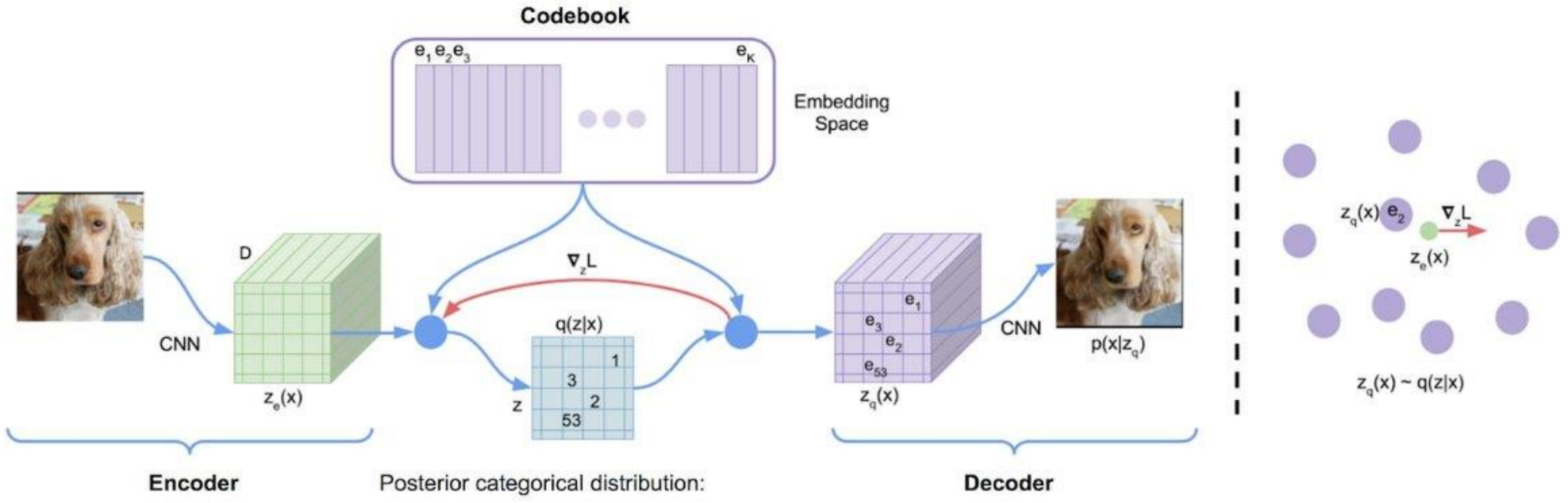
How to tokenize depth?



$$\xrightarrow{\quad} \langle \text{DEPTH\_}z_1^{\text{depth}} \rangle \ldots \langle \text{DEPTH\_}z_M^{\text{depth}} \rangle$$

Depth Perception Tokens

Vector Quantized VAE: Learn a discrete set of depth embeddings



Posterior categorical distribution:

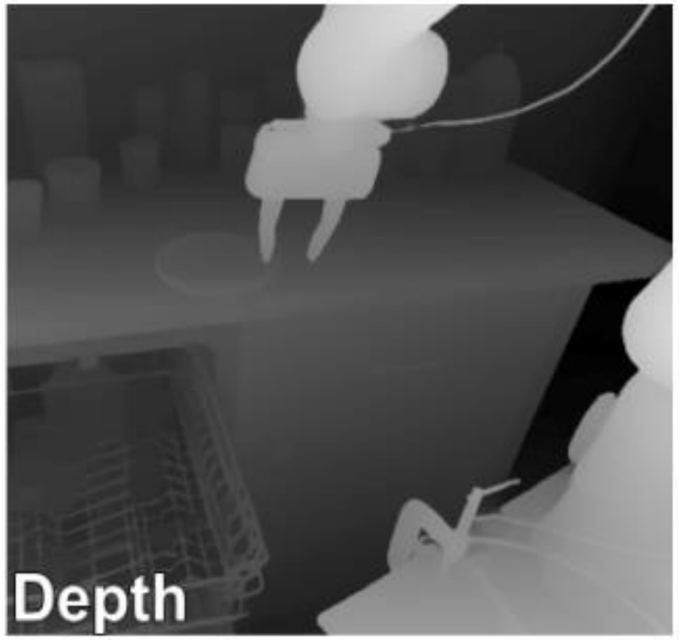
$$q(\mathbf{z} = \mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise}. \end{cases}$$

[5] Neural Discrete Representation Learning

Depth Perception Tokens

Idea: Predict Depth tokens from RGB Input

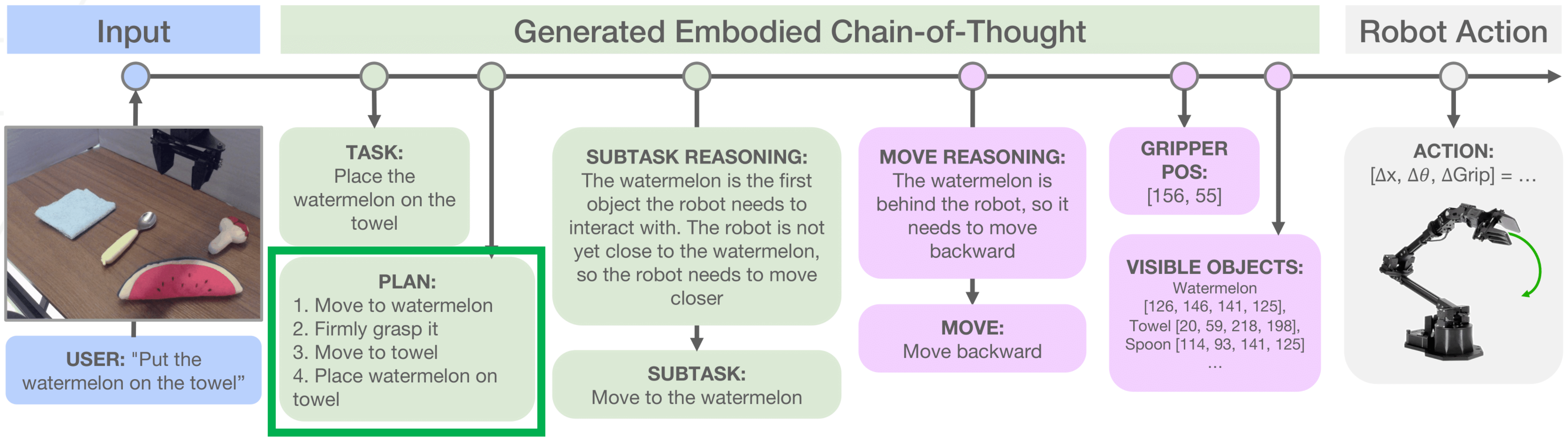
$\longrightarrow$



$$\langle \mathrm{DEPTH\_}z_1^{\mathrm{depth}} \rangle \ \ldots \ \langle \mathrm{DEPTH\_}z_M^{\mathrm{depth}} \rangle$$

Visual Reasoning Traces

Going beyond planning in language



USER: "Put the watermelon on the towel"

TASK: Place the watermelon on the towel

PLAN:
1. Move to watermelon
2. Firmly grasp it
3. Move to towel
4. Place watermelon on towel

SUBTASK REASONING: The watermelon is the first object the robot needs to interact with. The robot is not yet close to the watermelon, so the robot needs to move closer

SUBTASK: Move to the watermelon

MOVE REASONING: The watermelon is behind the robot, so it needs to move backward

MOVE: Move backward

GRIPPER POS: [156, 55]

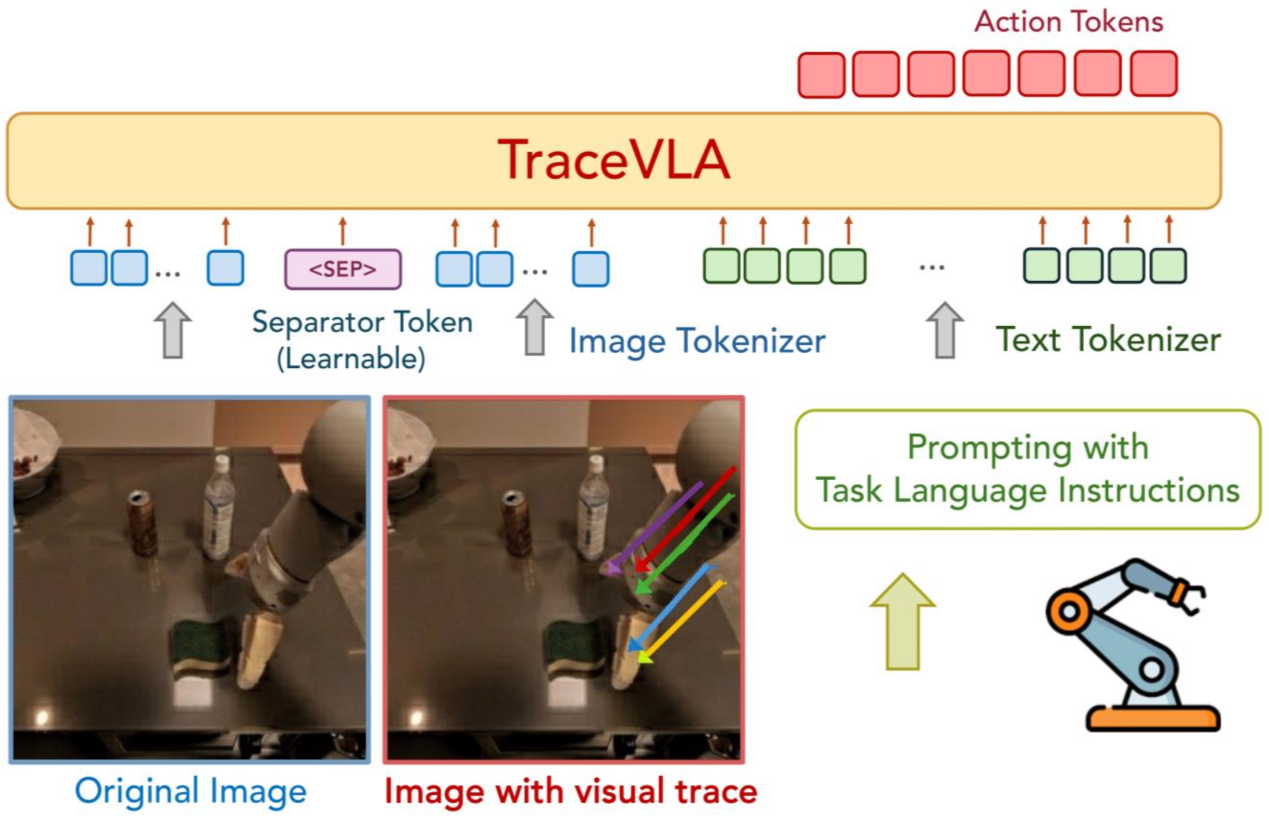
VISIBLE OBJECTS: Watermelon [126, 146, 141, 125], Towel [20, 59, 218, 198], Spoon [114, 93, 141, 125] …

ACTION: [Δx, $\Delta\theta$, ΔGrip] = …

Embodied Chain of Thought [2]

Visual Reasoning Traces

VLA's conditioned on 2d trajectories have been show to improve performance

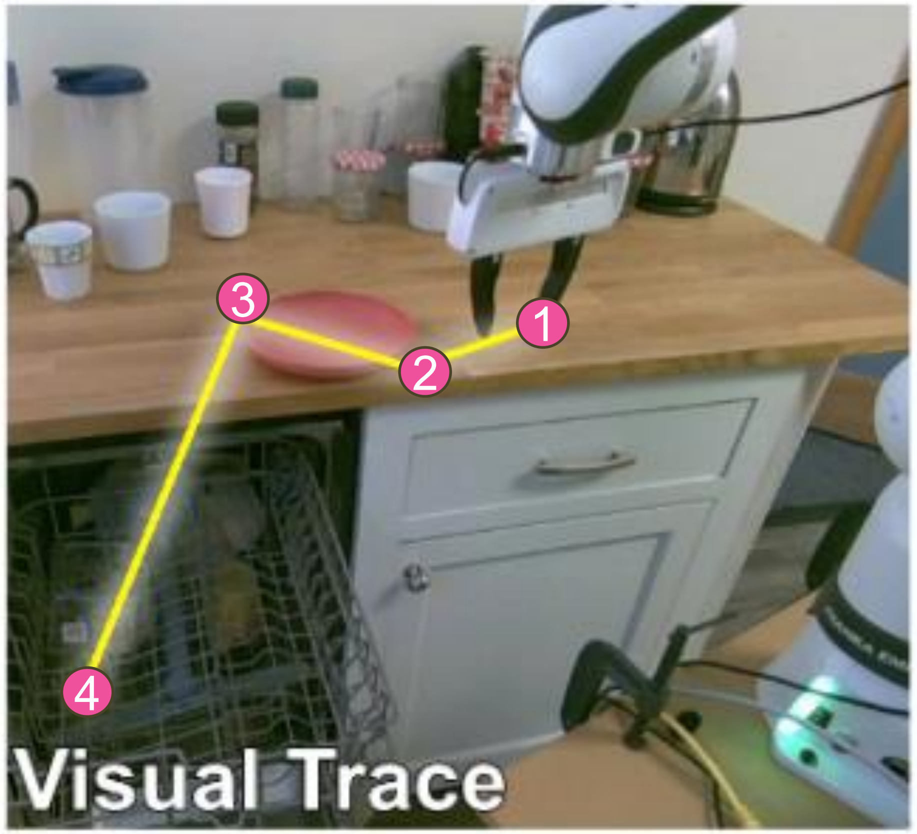


[6] Visual Trace Prompting Enhances Spatial-Temporal Awareness for Generalist Robotic Policies

Visual Reasoning Traces

Idea: train the model to predict the end effector trajectories.

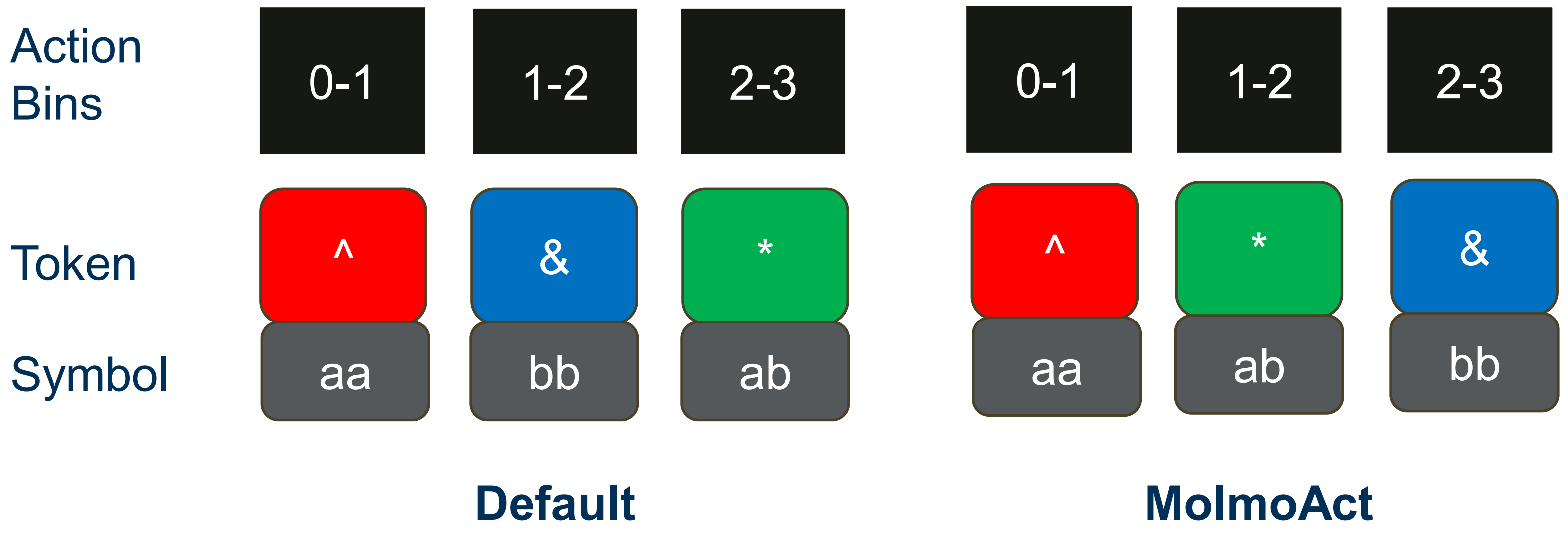
Predict a series of <u, v> coordinates in the image space corresponding to 3D end effector positions.



Action Tokens

Using Qwen2 tokenizer

1. Discretize distribution of actions seen in data into 256 bins
2. Take the last 256 tokens and order them based on their underlying BPE symbol
3. Assign each BPE symbol monotonically to an action bin



Full Action Reasoning Procedure

The task is Place the spoon on the lower left table edge. What is the action that the robot should take. To figure out the action that the robot should take to Place the spoon on the lower left table edge, let's think through it step by step. First, what is the depth map for this image? Second, what is the trajectory of the end effector? Based on the depth map of the image and the trajectory of the end effector, what is the action that the robot should take?

The depth map of the image is
‹DEPTH_START›‹DEPTH_75›‹DEPTH_101›‹DEPTH_69›‹DEPTH_8›‹DEPTH_69›‹DEPTH_56›‹DEPTH_56›‹DEPTH_30›‹
DEPTH_32›‹DEPTH_101›‹DEPTH_26›‹DEPTH_56›‹DEPTH_32›‹DEPTH_116›‹DEPTH_56›‹DEPTH_101›‹DEPTH_32›‹DEP
TH_8›‹DEPTH_69›‹DEPTH_8›‹DEPTH_75›‹DEPTH_32›‹DEPTH_116›‹DEPTH_9›‹DEPTH_9›‹DEPTH_116›‹DEPTH_32›‹
DEPTH_32›‹DEPTH_32›‹DEPTH_69›‹DEPTH_101›‹DEPTH_75›‹DEPTH_116›‹DEPTH_69›‹DEPTH_9›‹DEPTH_9›‹DEPTH
_9›‹DEPTH_69›‹DEPTH_9›‹DEPTH_101›‹DEPTH_69›‹DEPTH_69›‹DEPTH_69›‹DEPTH_69›‹DEPTH_69›‹DEPTH_69›
‹DEPTH_69›‹DEPTH_69›‹DEPTH_32›‹DEPTH_56›‹DEPTH_56›‹DEPTH_8›‹DEPTH_8›‹DEPTH_69›‹DEPTH_32›‹DEPT
H_56›‹DEPTH_32›‹DEPTH_9›‹DEPTH_56›‹DEPTH_56›‹DEPTH_56›‹DEPTH_56›‹DEPTH_8›‹DEPTH_32›‹DEPTH_32›
‹DEPTH_32›‹DEPTH_56›‹DEPTH_69›‹DEPTH_101›‹DEPTH_56›‹DEPTH_69›‹DEPTH_32›‹DEPTH_101›‹DEPTH_8›‹DEP
TH_56›‹DEPTH_101›‹DEPTH_69›‹DEPTH_69›‹DEPTH_75›‹DEPTH_101›‹DEPTH_69›‹DEPTH_25›‹DEPTH_74›‹DEPTH_
56›‹DEPTH_56›‹DEPTH_32›‹DEPTH_56›‹DEPTH_101›‹DEPTH_26›‹DEPTH_26›‹DEPTH_25›‹DEPTH_76›‹DEPTH_76›
‹DEPTH_76›‹DEPTH_103›‹DEPTH_69›‹DEPTH_25›‹DEPTH_74›‹DEPTH_116›‹DEPTH_116›‹DEPTH_END›. The trajectory of the end effector is [[30,39],[56,81],[58,70]]. Based on these information, the action that the robot should take is [áĬĴ, âķĬ, ðĲ¤Ķ, ì³ħ, ë±Ĳ, á¿Ĳ, ðŁĳ].

Sample CoT Prompt and response [3]

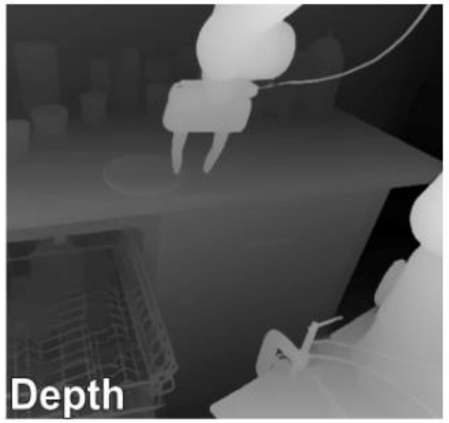
Full Action Reasoning Procedure

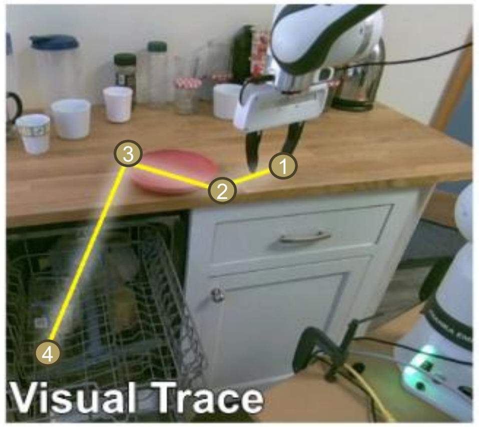
$$p(\mathbf{d}, \boldsymbol{\tau}, \mathbf{a} \,|\, I, T) = \prod_{i=1}^{M+2} p(d_i \mid I, T, \mathbf{d}_{<i}) \times \prod_{j=1}^{L} p(\tau_j \mid I, T, \mathbf{d}, \boldsymbol{\tau}_{<j}) \times \prod_{k=1}^{D} p(a_k \mid I, T, \mathbf{d}, \boldsymbol{\tau}, \mathbf{a}_{<k}).$$

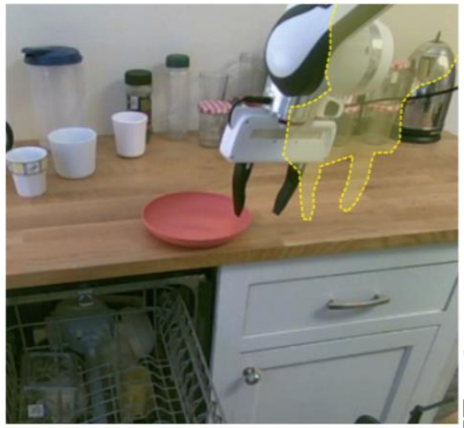
1 Predict depth map

2 Predict 2d trajectory

3 Predict actions

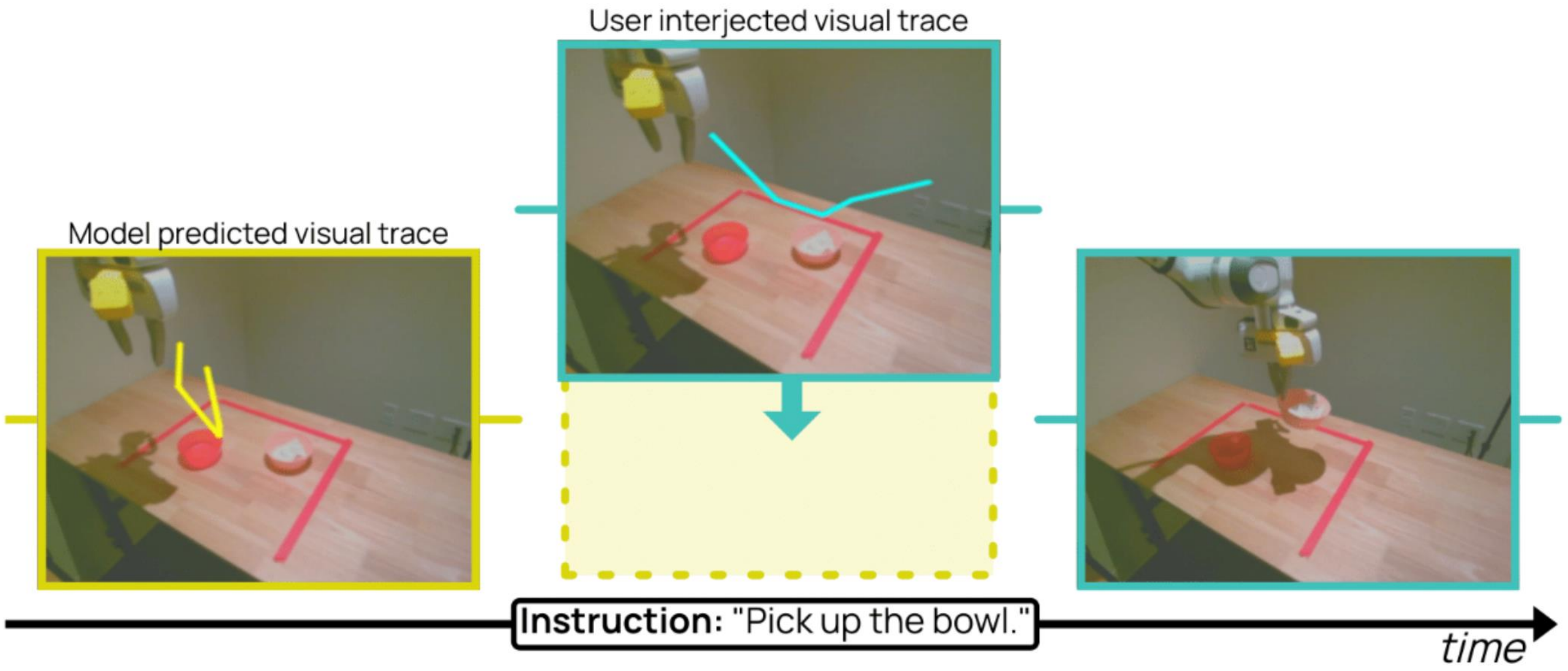






Incorporating Steerability

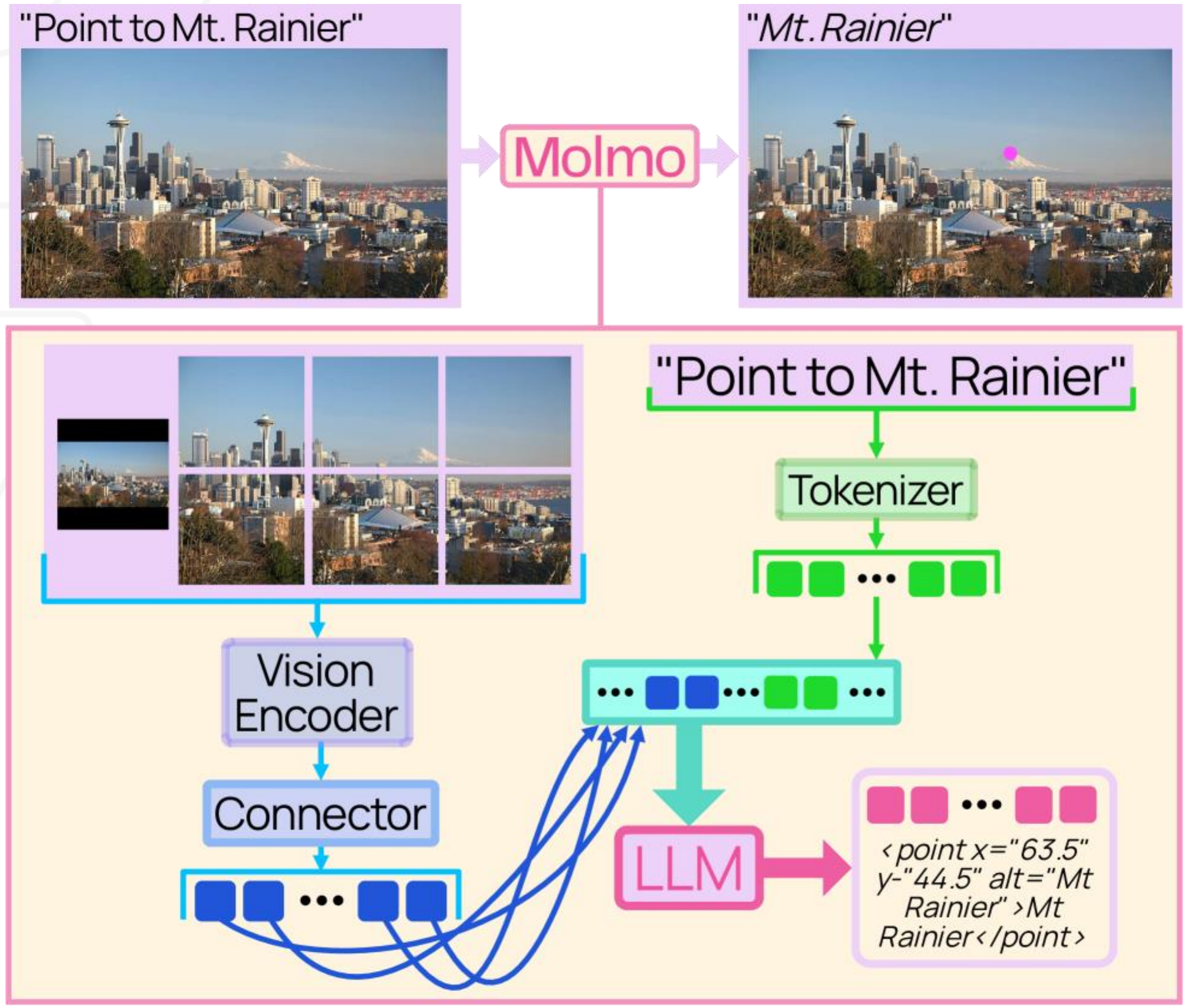
Idea: Encode desired end effector trajectories through visual reasoning traces



Instruction: "Pick up the bowl."

Training Action Reasoning Models

Model Architecture



Molmo [3]

Reuses Molmo Backbone

1. **Preprocessor**: Overlapping image crops with resized full resolution image
2. **ViT Image Encoder:** OpenAI ViT-L/14 336px CLIP, ViTSO400M/14 384px SigLIP2
3. **Vision-Language Connector:** 2 layer MLP
4. **LLM:** OLMo2-7B, Qwen2.5-7B

Training Stages

1	Pre-Training	Action Reasoning Data, Auxiliary Robot Data, Multimodal Web Data
2	Mid-Training	High Quality Action Reasoning Data
3	Post-Training	Small Scale Teleoperation Data

Georgia Tech

Pre-training Data Mixture

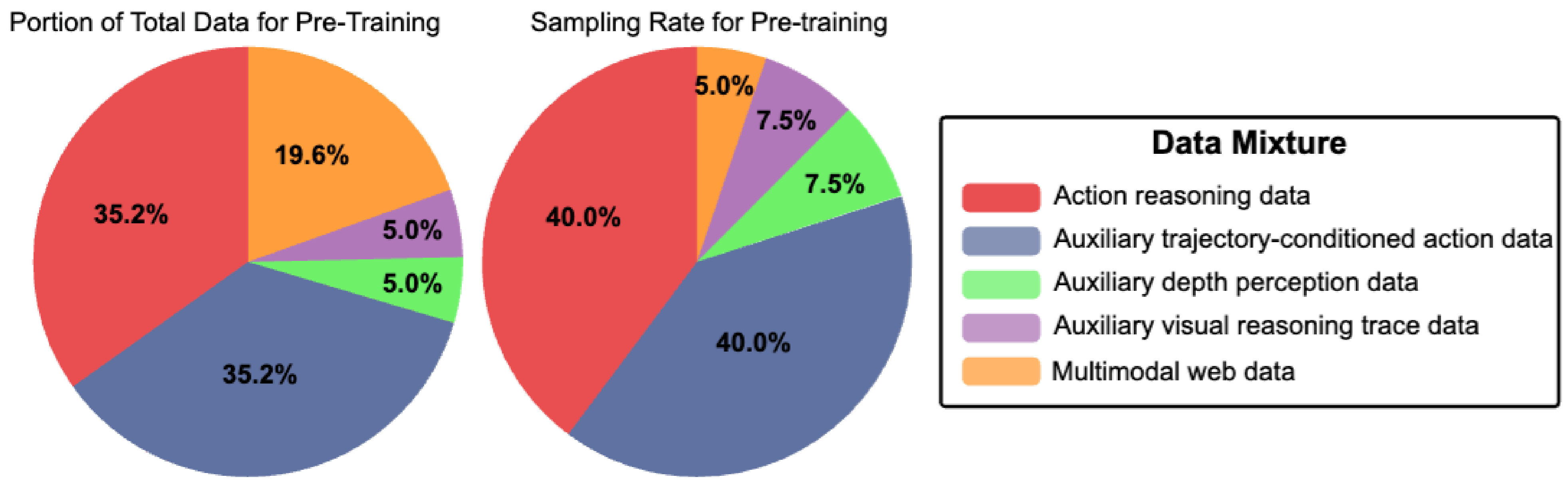
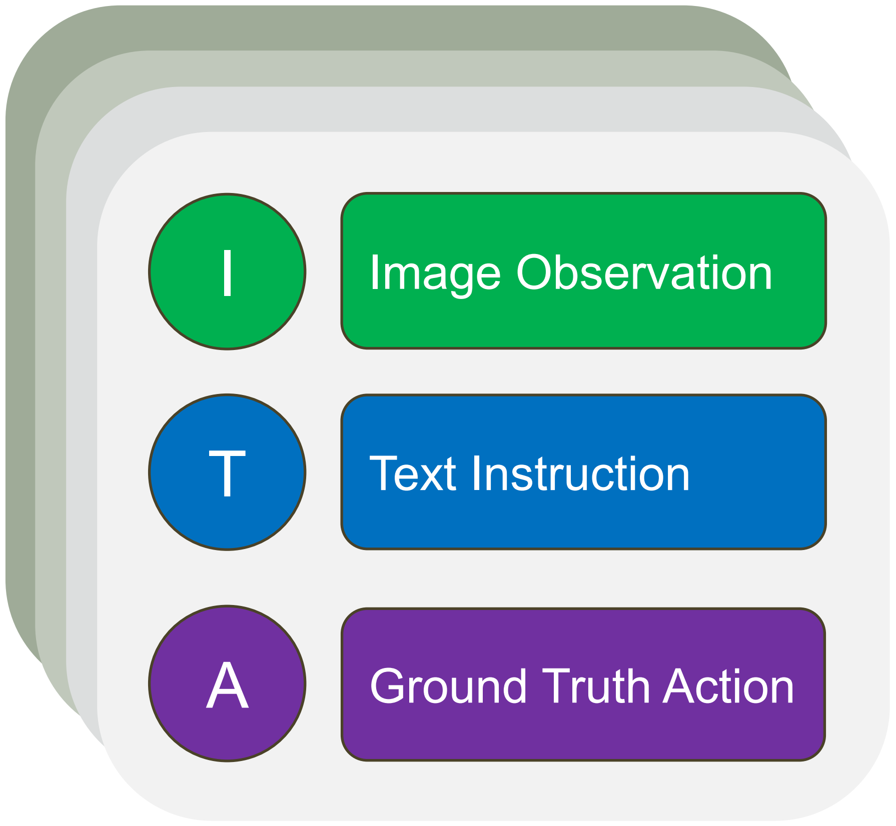


Figure 3 Distribution of data mixture in the overall pre-training mixture (left) and in the sampled subset used for MolmoAct pre-training (right). The mixture contains primarily action reasoning data (38.7%), trajectory-conditioned data (38.7%), and multimodal web data (21.5%), with small fractions of auxiliary depth and trace data (0.5% each). The sampled subset increases the proportion of auxiliary data (7.5% each for depth and line) while reducing multimodal web data to 5%.

Action Reasoning Data

Convert existing robot data to action reasoning data

Generally, robot data is a trajectory of tuples (I, T, A)



Action Reasoning Data

Obtaining depth tokens

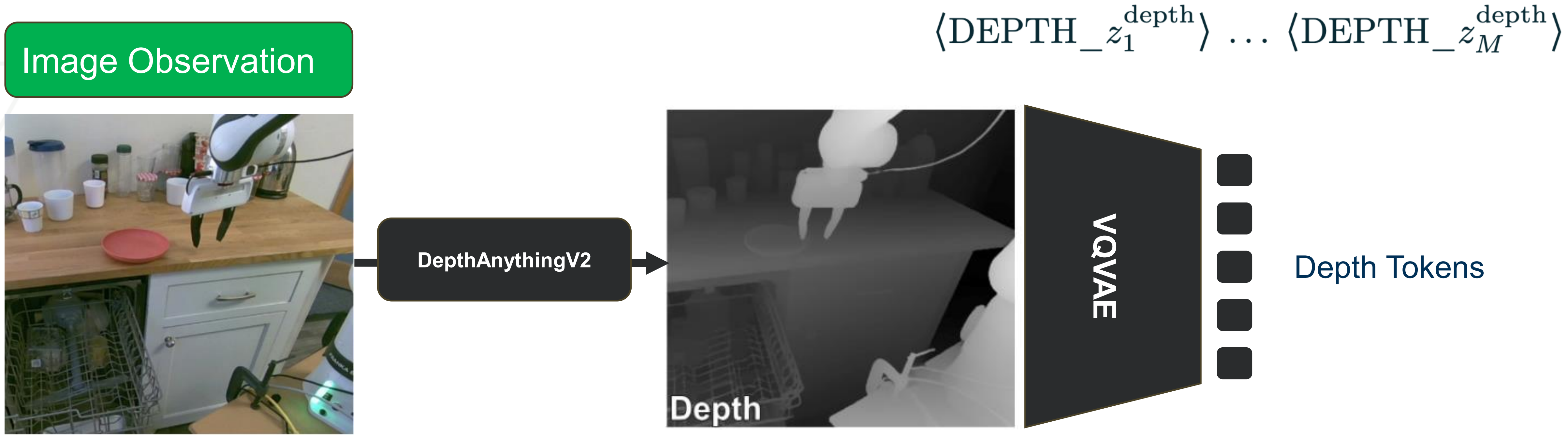
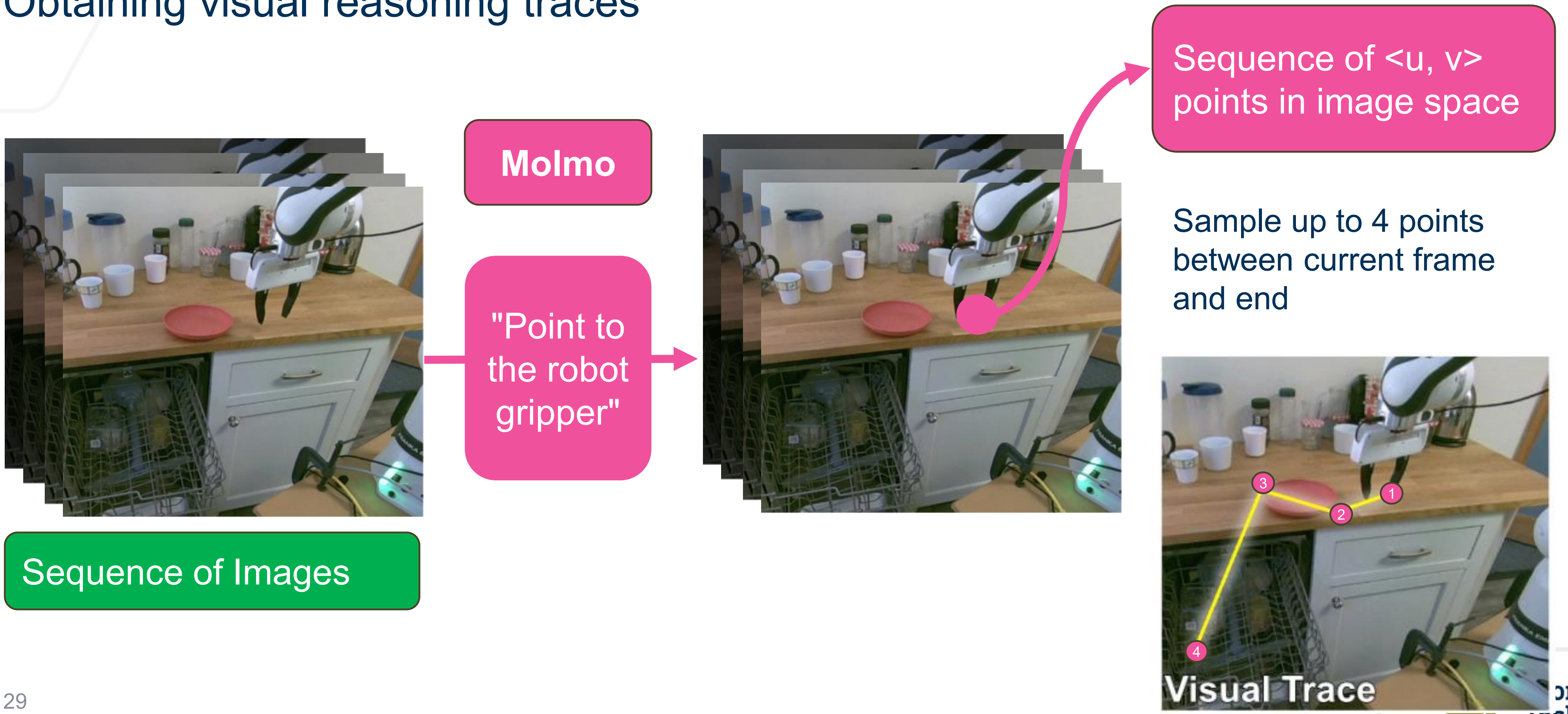


Image Observation
DepthAnythingV2
Depth
VQVAE
$\langle \text{DEPTH\_}z_1^{\text{depth}} \rangle \ldots \langle \text{DEPTH\_}z_M^{\text{depth}} \rangle$
Depth Tokens

Action Reasoning Data

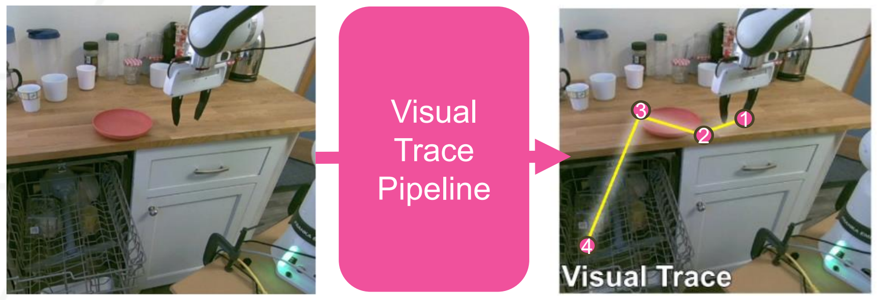
Obtaining visual reasoning traces


Sequence of Images
Molmo
"Point to the robot gripper"
Sequence of <u, v> points in image space
1
2
3
4
Visual Trace

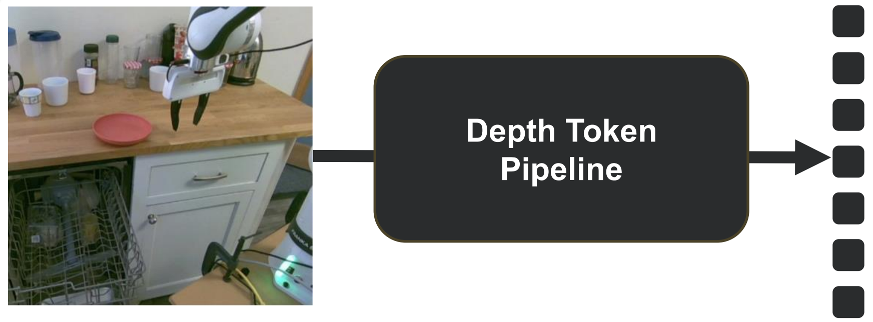
Sample up to 4 points between current frame and end

Auxiliary Robot Data

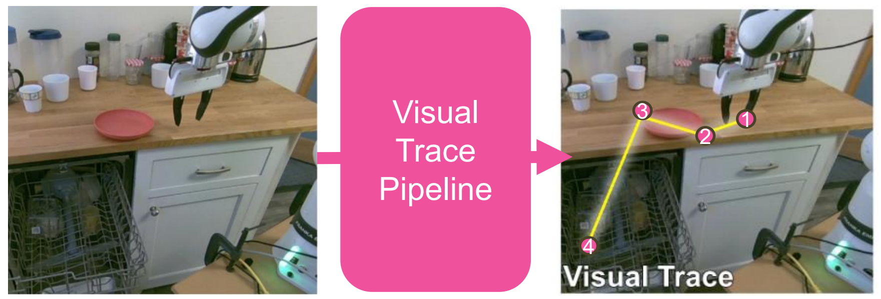
Use previously defined pipelines to create three new datasets



Auxiliary Trace Data: Given RGB observation and language instruction, predict corresponding visual trace

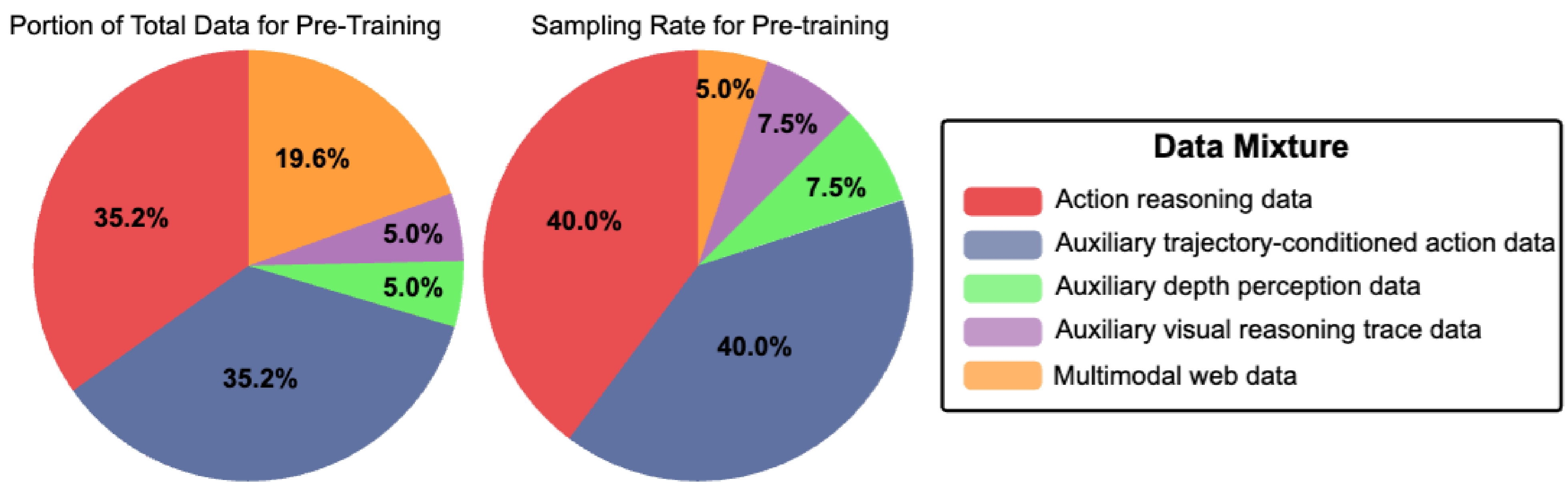


Auxiliary Depth Data: Given RGB observation and language instruction, predict target depth sequence



Trajectory Conditioned Action Data: Given a RGB observation, language instruction, and visual reasoning trace, predict the next action *(for steerability)*

Full Pre-training



10.5M subset of Open-X Embodiment data converted to action reasoning data

1.5M auxiliary depth data (sampled from

1.5M auxiliar trace data

10.5M trajectory conditioned action data

2M multimodal web data (taken from Molmo SFT mix)

Mid-training on the MolmoAct Dataset

10,689 Franka Panda trajectories, 93 tasks in home and table top environments

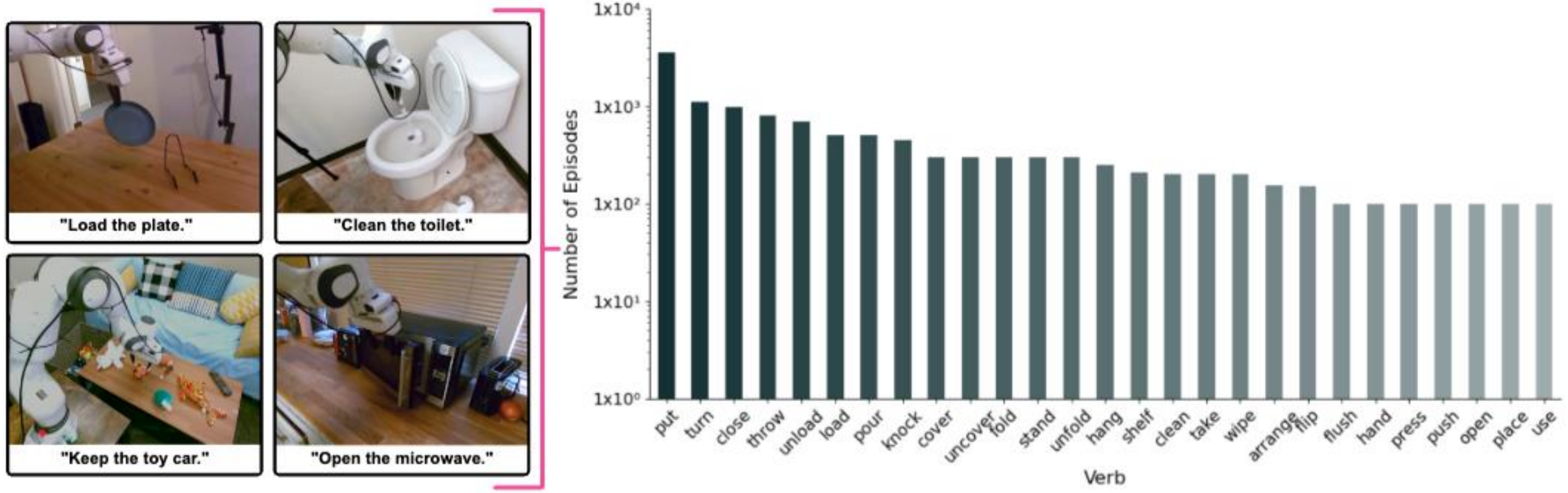


Figure 4 Examples and verb distribution in the MolmoAct Dataset. Left: Sample robot manipulation tasks paired with natural language instructions, spanning diverse household activities such as closing a laptop, loading a plate, cleaning a toilet, and opening a microwave. Right: Log-scale distribution of the top verbs in the dataset, showing a long-tail pattern with "put," "turn," and "close" as the most frequent actions.

Mid-training on the MolmoAct Dataset

1. Data is collected with two side cameras and a wrist mounted camera.
2. Each side view is paired with the wrist view

The task is clean the table. What is the action that the robot should take. To figure out the action that the robot should take to clean the table, let's think through it step by step.
First, what is the depth map for the first image?
Second, what is the trajectory of the end effector in the first image?
Based on the depth map of the first image and the trajectory of the end effector in the first image, along with other images from different camera views as additional information, what is the action that the robot should take?

1M action reasoning data samples, 1M trajectory conditioned action data samples

Post-training

For a given task

1. Collect 30-50 teleop demos
2. Convert to Action Reasoning Data and Trajectory Conditioned Action Data
3. Apply action chunking (label next 8 actions instead of just 1)
4. LoRA finetuning

Evaluation

In Distribution

- Evaluated pre-training only model on SimpleEnv to isolate in-distribution performance and pre-trained capabilities and evaluate out-of-box generalization
- Models finetuned on RT-1 subset were evaluated, negligible improvement suggests pre-trained MolmoAct is strong initialization for mid-training onwards
- Pre-training distribution (OXE subset) is most similar to Google Robot visual-matching task in SimpleEnv (RT-1 subset may align more given fine-tuning)
- Compared against prior VLAs trained on private data + full OXE data
    - Pre-trained MolmoAct only trains on subset of OXE: 26.3M vs 903M
- Most baseline models were evaluated zero-shot

In Distribution

- Evaluated pre-training only model on SimpleEnv to isolate in-distribution performance and pre-trained capabilities and evaluate out-of-box generalization
- Models finetuned on RT-1 subset were evaluated, negligible improvement suggests pre-trained MolmoAct is strong initialization for mid-training onwards

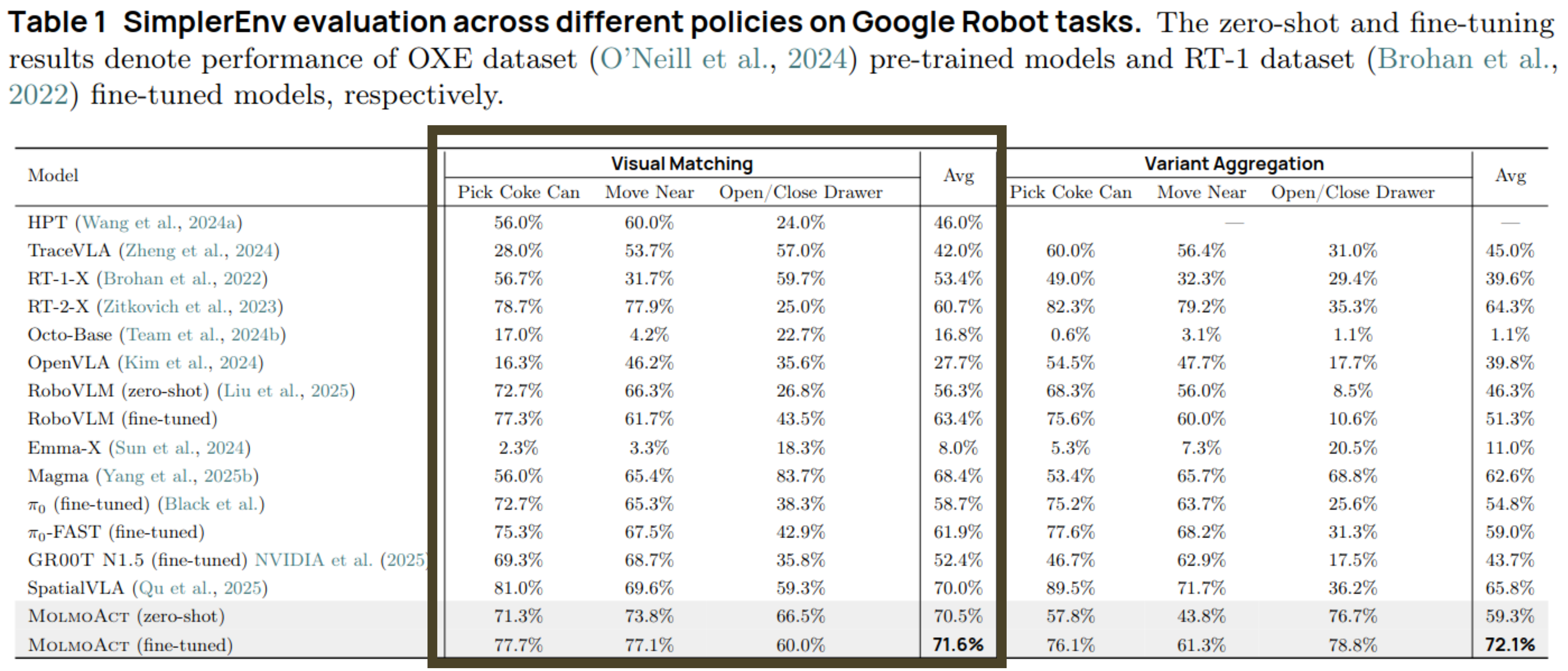


Table 1 SimplerEnv evaluation across different policies on Google Robot tasks. The zero-shot and fine-tuning results denote performance of OXE dataset (O'Neill et al., 2024) pre-trained models and RT-1 dataset (Brohan et al., 2022) fine-tuned models, respectively.

Model	Visual Matching			Avg	Variant Aggregation			Avg
	Pick Coke Can	Move Near	Open/Close Drawer		Pick Coke Can	Move Near	Open/Close Drawer	
HPT (Wang et al., 2024a)	56.0%	60.0%	24.0%	46.0%		—		—
TraceVLA (Zheng et al., 2024)	28.0%	53.7%	57.0%	42.0%	60.0%	56.4%	31.0%	45.0%
RT-1-X (Brohan et al., 2022)	56.7%	31.7%	59.7%	53.4%	49.0%	32.3%	29.4%	39.6%
RT-2-X (Zitkovich et al., 2023)	78.7%	77.9%	25.0%	60.7%	82.3%	79.2%	35.3%	64.3%
Octo-Base (Team et al., 2024b)	17.0%	4.2%	22.7%	16.8%	0.6%	3.1%	1.1%	1.1%
OpenVLA (Kim et al., 2024)	16.3%	46.2%	35.6%	27.7%	54.5%	47.7%	17.7%	39.8%
RoboVLM (zero-shot) (Liu et al., 2025)	72.7%	66.3%	26.8%	56.3%	68.3%	56.0%	8.5%	46.3%
RoboVLM (fine-tuned)	77.3%	61.7%	43.5%	63.4%	75.6%	60.0%	10.6%	51.3%
Emma-X (Sun et al., 2024)	2.3%	3.3%	18.3%	8.0%	5.3%	7.3%	20.5%	11.0%
Magma (Yang et al., 2025b)	56.0%	65.4%	83.7%	68.4%	53.4%	65.7%	68.8%	62.6%
π_0 (fine-tuned) (Black et al.)	72.7%	65.3%	38.3%	58.7%	75.2%	63.7%	25.6%	54.8%
π_0-FAST (fine-tuned)	75.3%	67.5%	42.9%	61.9%	77.6%	68.2%	31.3%	59.0%
GR00T N1.5 (fine-tuned) NVIDIA et al. (2025	69.3%	68.7%	35.8%	52.4%	46.7%	62.9%	17.5%	43.7%
SpatialVLA (Qu et al., 2025)	81.0%	69.6%	59.3%	70.0%	89.5%	71.7%	36.2%	65.8%
MolmoAct (zero-shot)	71.3%	73.8%	66.5%	70.5%	57.8%	43.8%	76.7%	59.3%
MolmoAct (fine-tuned)	77.7%	77.1%	60.0%	**71.6%**	76.1%	61.3%	78.8%	**72.1%**

Finetuning

- Finetuned MolmoAct via post-training (after mid-training) using LoRA on LIBERO to evaluate adaptability and compare against generalist autoregressive policies
- MolmoAct does match or outperform the baselines, but it seems unfair to compare against generalist policies without finetuning them as well?

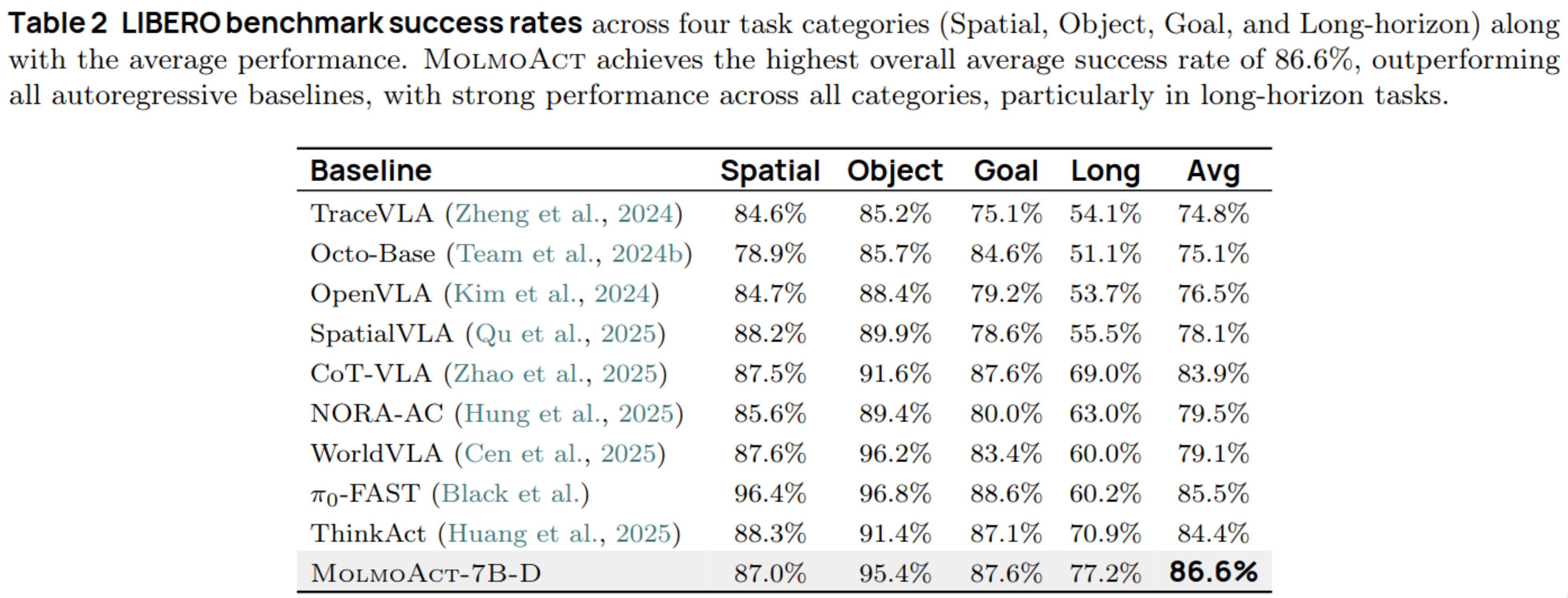


Table 2 LIBERO benchmark success rates across four task categories (Spatial, Object, Goal, and Long-horizon) along with the average performance. MolmoAct achieves the highest overall average success rate of 86.6%, outperforming all autoregressive baselines, with strong performance across all categories, particularly in long-horizon tasks.

Baseline	Spatial	Object	Goal	Long	Avg
TraceVLA (Zheng et al., 2024)	84.6%	85.2%	75.1%	54.1%	74.8%
Octo-Base (Team et al., 2024b)	78.9%	85.7%	84.6%	51.1%	75.1%
OpenVLA (Kim et al., 2024)	84.7%	88.4%	79.2%	53.7%	76.5%
SpatialVLA (Qu et al., 2025)	88.2%	89.9%	78.6%	55.5%	78.1%
CoT-VLA (Zhao et al., 2025)	87.5%	91.6%	87.6%	69.0%	83.9%
NORA-AC (Hung et al., 2025)	85.6%	89.4%	80.0%	63.0%	79.5%
WorldVLA (Cen et al., 2025)	87.6%	96.2%	83.4%	60.0%	79.1%
π_0-FAST (Black et al.)	96.4%	96.8%	88.6%	60.2%	85.5%
ThinkAct (Huang et al., 2025)	88.3%	91.4%	87.1%	70.9%	84.4%
MolmoAct-7B-D	87.0%	95.4%	87.6%	77.2%	**86.6%**

Finetuning

- Finetuned MolmoAct via post-training (after mid-training) using LoRA on real-world tasks to evaluate adaptability and compare against autoregressive policies
- MolmoAct does match or outperform the baselines, but it seems unfair to compare against generalist policies without finetuning them as well?

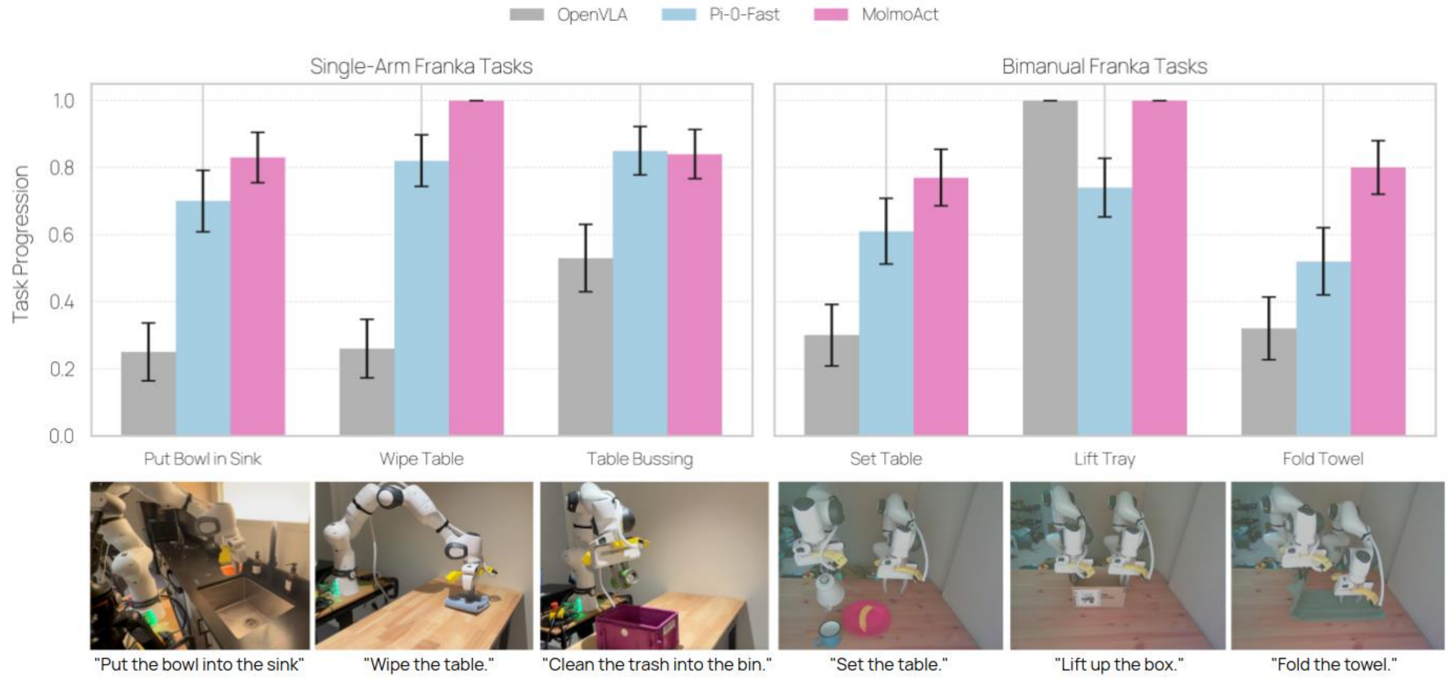


Figure 5 Real-world evaluation of OpenVLA, π_0-FAST, and MolmoAct on single-arm (left) and bimanual (right) Franka tasks. Bar plots report average task progression with standard error across 25 trials per task. MolmoAct consistently outperforms baselines, particularly on single-arm tasks such as *Wipe Table* and *Table Bussing*, and maintains strong performance on bimanual tasks including *Fold Towel* and *Set Table*. Bottom row shows example task setups with corresponding natural language instructions.

OOD Generalization

- Evaluated pre-training only model on SimpleEnv unseen variants: language variation, spatial variation, distractors, novel objects
- Models finetuned on RT-1 subset were evaluated, fine-tuned MolmoAct outperforms all baselines by a margin of at least 7.8%

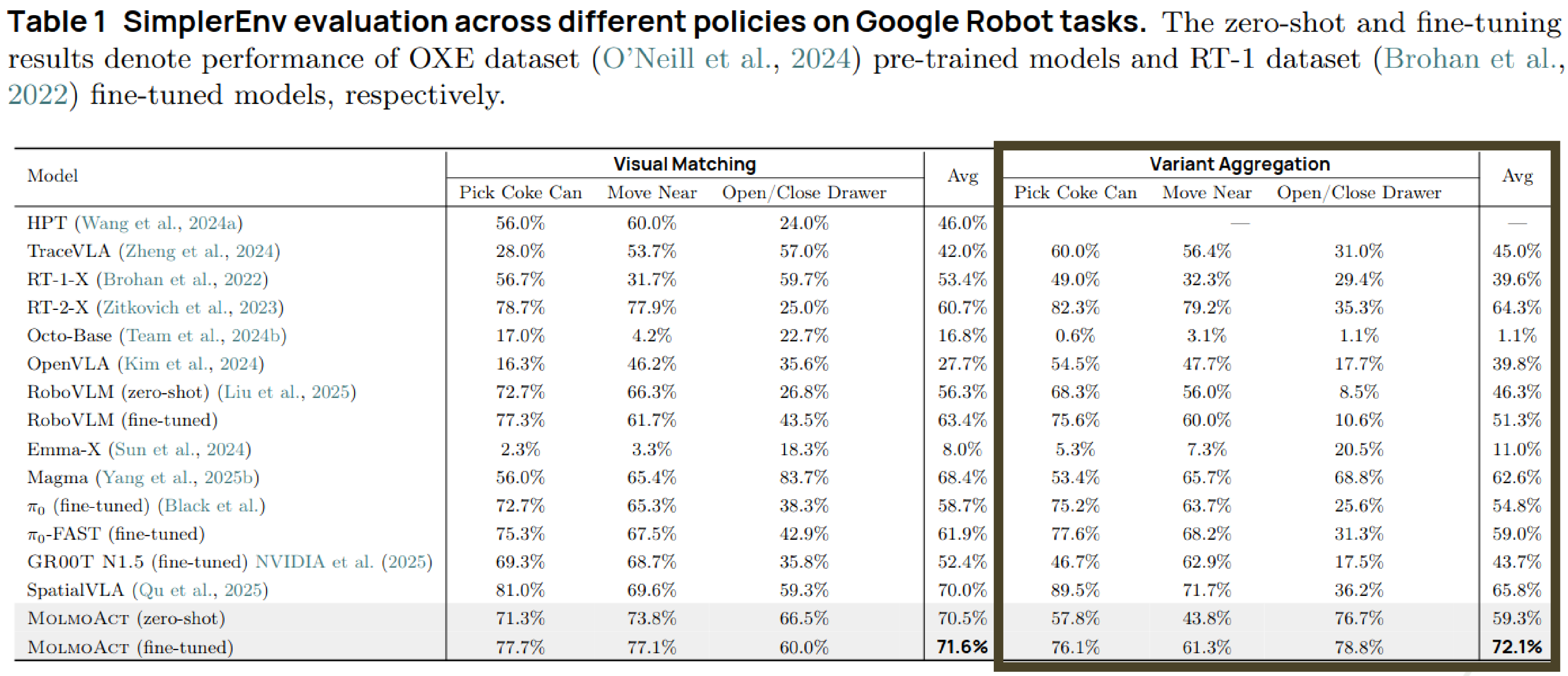
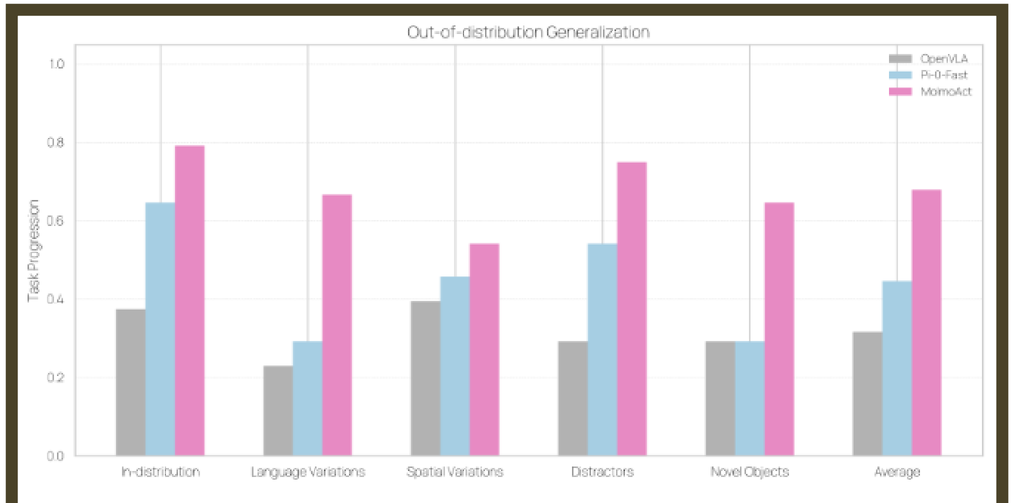


Table 1 SimplerEnv evaluation across different policies on Google Robot tasks. The zero-shot and fine-tuning results denote performance of OXE dataset (O'Neill et al., 2024) pre-trained models and RT-1 dataset (Brohan et al., 2022) fine-tuned models, respectively.

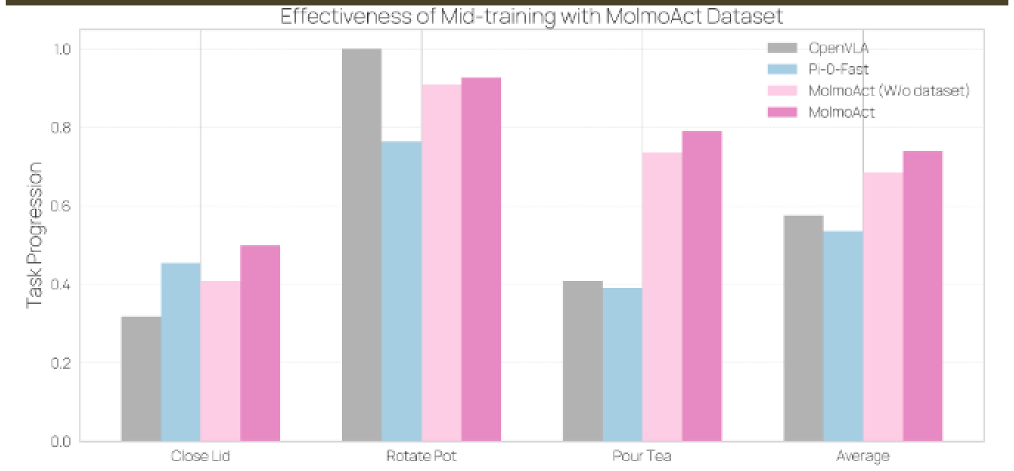
Model	Visual Matching			Avg	Variant Aggregation			Avg
	Pick Coke Can	Move Near	Open/Close Drawer		Pick Coke Can	Move Near	Open/Close Drawer	
HPT (Wang et al., 2024a)	56.0%	60.0%	24.0%	46.0%		—		—
TraceVLA (Zheng et al., 2024)	28.0%	53.7%	57.0%	42.0%	60.0%	56.4%	31.0%	45.0%
RT-1-X (Brohan et al., 2022)	56.7%	31.7%	59.7%	53.4%	49.0%	32.3%	29.4%	39.6%
RT-2-X (Zitkovich et al., 2023)	78.7%	77.9%	25.0%	60.7%	82.3%	79.2%	35.3%	64.3%
Octo-Base (Team et al., 2024b)	17.0%	4.2%	22.7%	16.8%	0.6%	3.1%	1.1%	1.1%
OpenVLA (Kim et al., 2024)	16.3%	46.2%	35.6%	27.7%	54.5%	47.7%	17.7%	39.8%
RoboVLM (zero-shot) (Liu et al., 2025)	72.7%	66.3%	26.8%	56.3%	68.3%	56.0%	8.5%	46.3%
RoboVLM (fine-tuned)	77.3%	61.7%	43.5%	63.4%	75.6%	60.0%	10.6%	51.3%
Emma-X (Sun et al., 2024)	2.3%	3.3%	18.3%	8.0%	5.3%	7.3%	20.5%	11.0%
Magma (Yang et al., 2025b)	56.0%	65.4%	83.7%	68.4%	53.4%	65.7%	68.8%	62.6%
π_0 (fine-tuned) (Black et al.)	72.7%	65.3%	38.3%	58.7%	75.2%	63.7%	25.6%	54.8%
π_0-FAST (fine-tuned)	75.3%	67.5%	42.9%	61.9%	77.6%	68.2%	31.3%	59.0%
GR00T N1.5 (fine-tuned) NVIDIA et al. (2025)	69.3%	68.7%	35.8%	52.4%	46.7%	62.9%	17.5%	43.7%
SpatialVLA (Qu et al., 2025)	81.0%	69.6%	59.3%	70.0%	89.5%	71.7%	36.2%	65.8%
MolmoAct (zero-shot)	71.3%	73.8%	66.5%	70.5%	57.8%	43.8%	76.7%	59.3%
MolmoAct (fine-tuned)	77.7%	77.1%	60.0%	**71.6%**	76.1%	61.3%	78.8%	**72.1%**

OOD Generalization

- Evaluated mid-trained model on unseen real-world multi-task setup
- Models finetuned on over 300 teleoperated demonstrations and evaluated on variants: language variation, spatial variation, distractors, novel objects
- Average improvement of MolmoAct over best baseline (Pi-0 Fast) was 23.3%



(a) MolmoAct generalizes beyond training distributions.

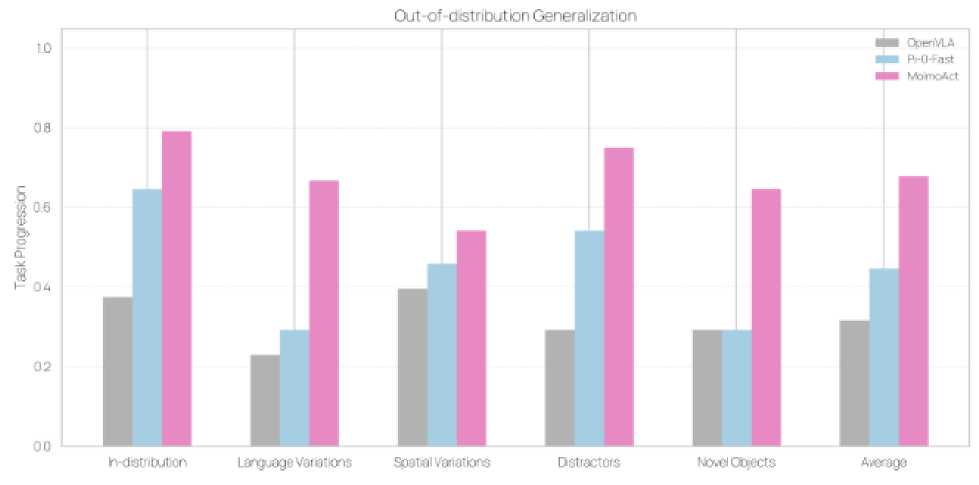


(b) MolmoAct Dataset improves task performance.

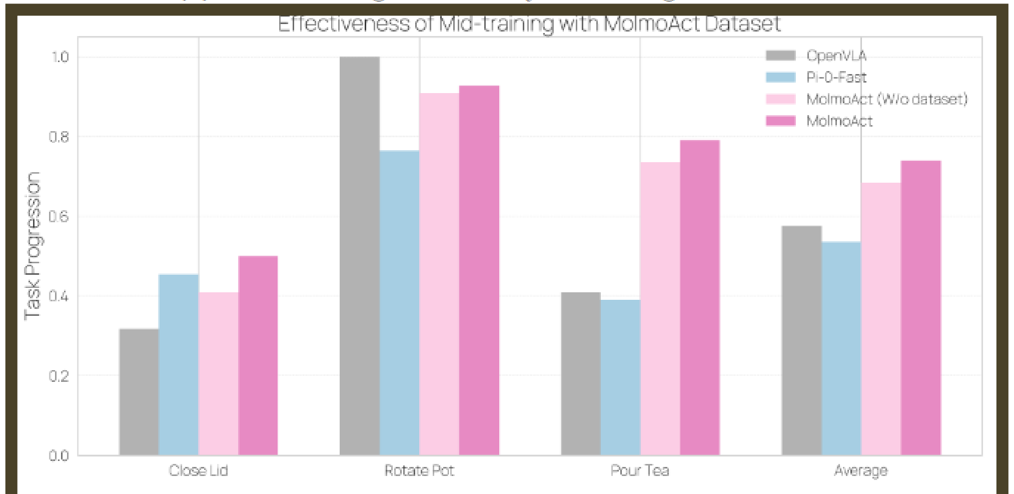
Figure 6 MolmoAct outperforms baselines across generalization and mid-training settings. (a) Out-of-distribution generalization: Task progression scores for OpenVLA, π_0-FAST, and MolmoAct across in-distribution, language variation, spatial variation, distractors, and novel object conditions, showing consistent gains for MolmoAct. (b) Effectiveness of mid-training with the MolmoAct Dataset: Comparison of task progression on real-world tasks (Close Lid, Rotate Pot, Pour Tea) for MolmoAct with and without the dataset, π_0-FAST, and OpenVLA, demonstrating that mid-training with the dataset improves performance across tasks.

Impact of Mid-Training

- Evaluated mid-training efficacy on three real-world tasks beyond pick-and-place
- Finetuned MolmoAct prior versus post mid-training along with finetuning two other baseline models
- Mid-training improves average performance by 5%, and both models generally perform better than baselines (exception: rotate pot)



(a) MOLMOACT generalizes beyond training distributions.

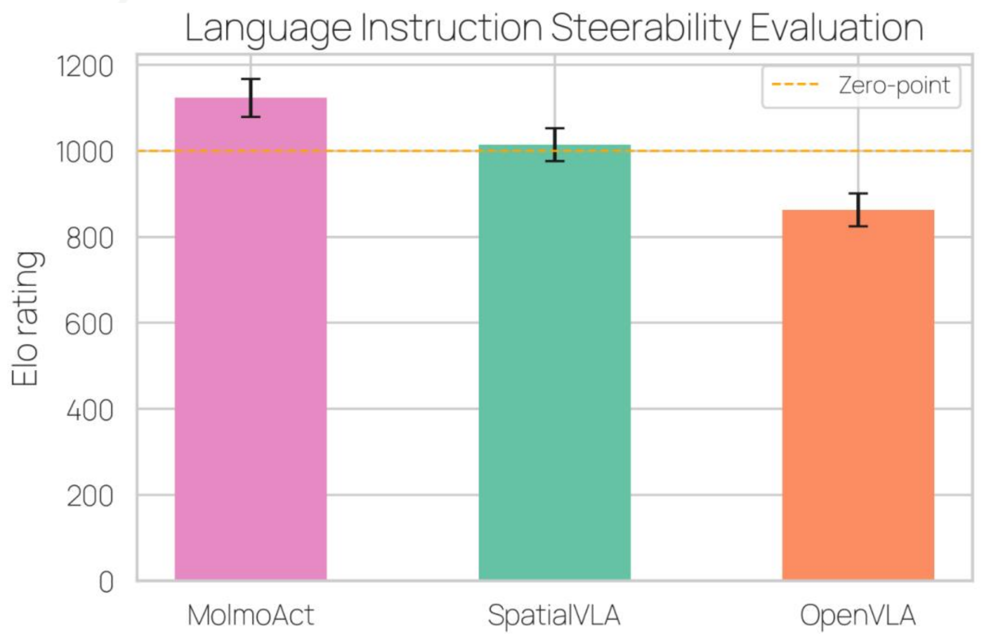


(b) MOLMOACT DATASET improves task performance.

Figure 6 MolmoAct outperforms baselines across generalization and mid-training settings. (a) Out-of-distribution generalization: Task progression scores for OpenVLA, π_0-FAST, and MOLMOACT across in-distribution, language variation, spatial variation, distractors, and novel object conditions, showing consistent gains for MOLMOACT. (b) Effectiveness of mid-training with the MOLMOACT DATASET: Comparison of task progression on real-world tasks (Close Lid, Rotate Pot, Pour Tea) for MOLMOACT with and without the dataset, π_0-FAST, and OpenVLA, demonstrating that mid-training with the dataset improves performance across tasks.

Language Following

- Conducted head-to-head arena-style evaluation with human evaluators (n=100)
- Generated rollouts for method pairs, 1500 votes collected, converted to Elo score
- Evaluated pre-trained MolmoAct against two VLAs pre-trained on OXE dataset
- Evaluated on five manipulation scenarios only, selection criteria unknown



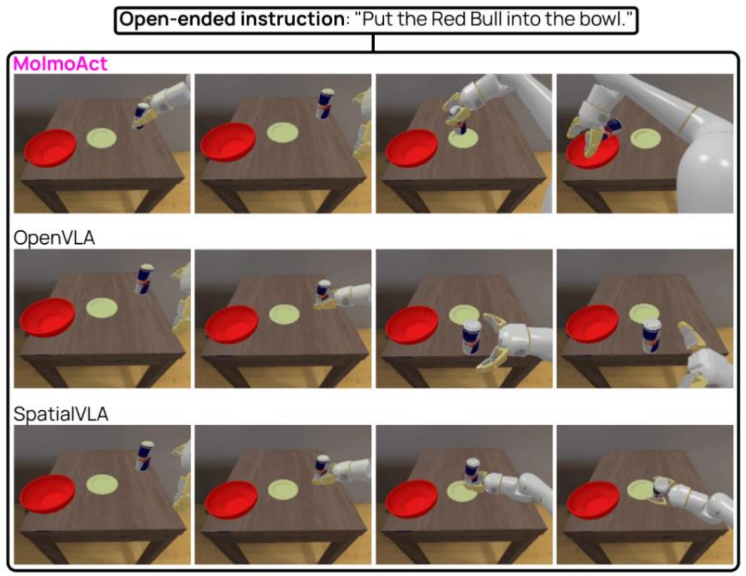
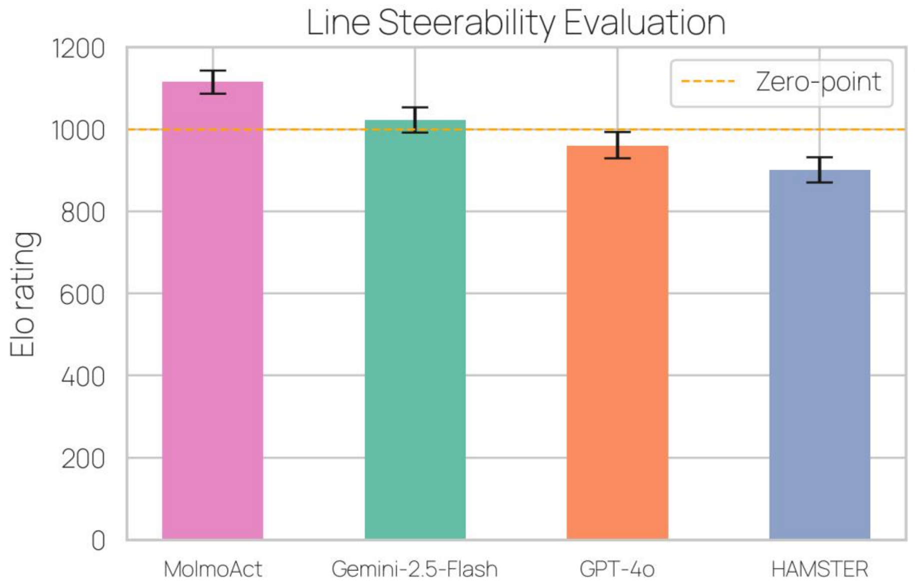


Figure 18 Language Instruction Evaluation. Left: Elo ratings for three models based on human votes in a head-to-head instruction-following evaluation. **Right:** Qualitative comparison of execution traces for the open-ended instruction "Put the redbull into the bowl." MolmoAct aligns more closely with the intended instruction than other models.

Language Following

- Conducted head-to-head arena-style evaluation with human evaluators (n=100)
- Generated traces for method pairs, 1500 votes collected, converted to Elo score
- Evaluated pre-trained MolmoAct against VLMs, including one for trace generation
- Interestingly, HAMSTER (trace generation VLA) performs worst



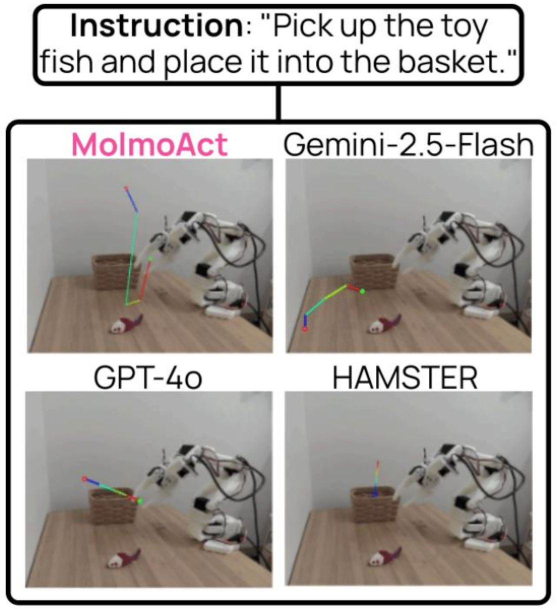
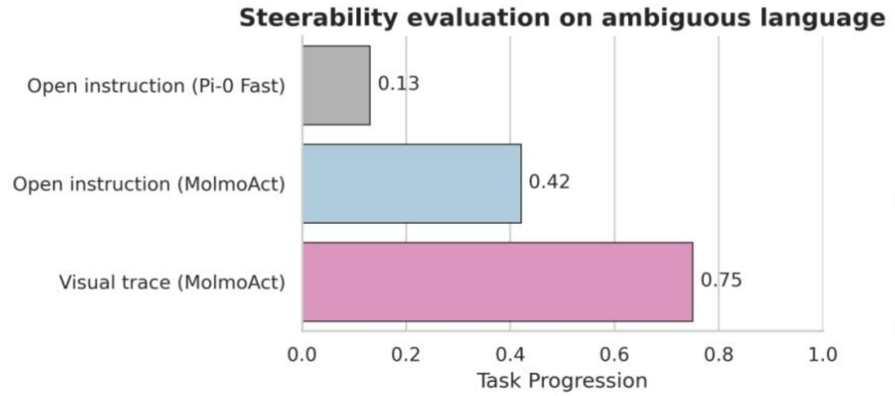

Instruction: "Pick up the toy fish and place it into the basket."
MolmoAct
Gemini-2.5-Flash
GPT-4o
HAMSTER

Figure 7 **Line steerability evaluation across models.** **Left:** Elo ratings show that MolmoAct achieves the highest performance, surpassing Gemini-2.5-Flash, GPT-4o, and HAMSTER, with error bars indicating 95% confidence interval (CI). **Right:** Example qualitative results showing predicted visual traces overlaid on robot camera views.

Steerability

- Evaluate ability to steer VLA and mediums for steering VLAs (trace vs text)
- Use ambiguous instruction to generate initial trajectory that user intervenes in
- Evaluate MolmoAct after mid-training versus baseline VLA model on 10 users
- Interesting that MolmoAct outperforms Pi-0 Fast in open instruction



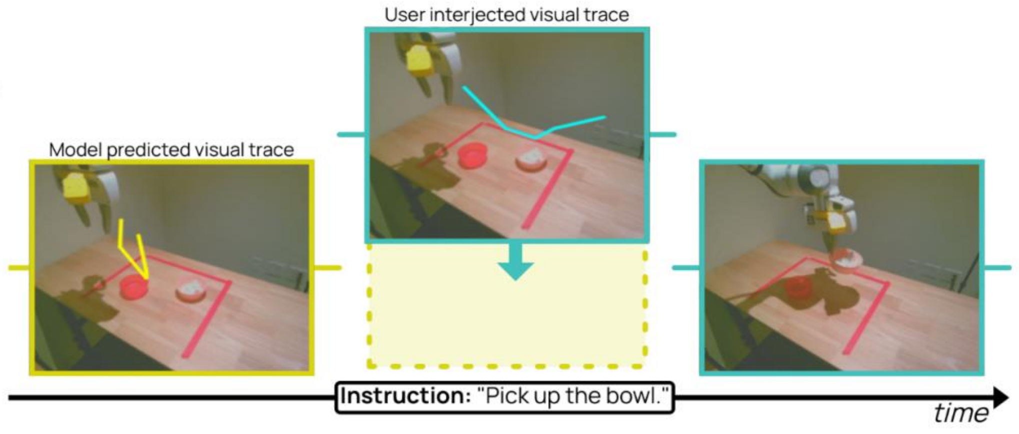

User interjected visual trace
Model predicted visual trace
Instruction: "Pick up the bowl."
time

Figure 9 Steerability evaluation with open instructions and visual traces. Left: Success rates for different steering modes, showing that MolmoAct with visual trace steering achieves the highest success rate (0.75), outperforming its open-instruction variant and π_0-FAST. **Right:** Example of the *"Pick up the bowl"* task: the model-predicted trajectory (yellow) is adjusted via a user-provided steering trajectory (cyan), resulting in the corrected task completion.

Limitations

Real-Time Execution

- Inference is conducted on server, so control frequency during data collection does not match the control frequency during inference due to latency
- Larger inference times due to latency and need to predict large number of reasoning tokens implies that control frequency realized is likely low
- Only one LLM parameter size (7B) tested (OLMo2-7B or Qwen2.5-7B), so unclear whether smaller models could be a solution without sacrificing performance
- Note that all main paper results use the Qwen2.5-7B LLM backbone despite having tested on OLMo2-7B backbone (which is their most open model)

Perception-Related Issues

- Spatial reasoning is primarily done through front camera despite having access to multiple camera views (front and wrist cameras)
- Front camera produces view of end-effector for most scenarios, and view of end-effector is needed for visual trace prediction and action prediction (eventually)
- When front camera is occluded, trace prediction and action prediction degrade
- Authors propose wide field-of-view camera and generate visual traces via SLAM
- Approaches in occlusion robustness for visual tracking could be used?
- Fixed set of 100 tokens for representing depth is hand-chosen, but certain (fine-grained) tasks may require more representation for depth estimation

Robustness of Steering via Visual Traces

- MolmoAct leverages a trace superimposed onto 2D image, so there is inherently no 3D or depth encoding in the visual trace itself (cue is primarily 2D)
- This can be okay when training (as shown by results), but when intervention occurs, if the trace is not similar to data distribution, it may fail to follow it
- Options:
  - Various traces should map to various action compositions to better improve trace-action mapping coverage (essentially more diverse trace-action data to improve support)
  - Condition on or reuse depth information along with the trace simultaneously to introduce 3D or depth information back (as opposed to sequential decomposition as is currently done)

Summary of Strengths, Weaknesses, Relationships

- Simple three-stage reasoning structure admits explainable and steerable VLA behavior
- Method performs well compared to baselines both in task performance and intuitiveness of explanations
- Fully (essentially) open model, including model parameters, source code, and training dataset

- Slow inference and execution time and lack of study into smaller parameter backbones and models
- Three-stage approach is inductive bias with failure cases and modes
- Results only presented for one of the two variants proposed by authors
- Only tabletop manipulation tasks considered in this study

Discussion Questions

- Would the inductive structure that was introduced in MolmoAct transpose well to other domains (locomotion, whole-body control, etc.)?

Discussion Questions

- Would the inductive structure that was introduced in MolmoAct transpose well to other domains (locomotion, whole-body control, etc.)?
- In addition to depth perception and visual reasoning traces, what are other methods of chain of thought prompting to better ground VLAs?

Discussion Questions

- Would the inductive structure that was introduced in MolmoAct transpose well to other domains (locomotion, whole-body control, etc.)?
- In addition to depth perception and visual reasoning traces, what are other methods of chain of thought prompting to better ground VLAs?
- This paper proposes 2d drawings as an interface for steering VLAs? Is this the best way to steer VLAs?

Discussion Questions

- Would the inductive structure that was introduced in MolmoAct transpose well to other domains (locomotion, whole-body control, etc.)?
- In addition to depth perception and visual reasoning traces, what are other methods of chain of thought prompting to better ground VLAs?
- This paper proposes 2d drawings as an interface for steering VLAs? Is this the best way to steer VLAs?
- Is self-explainability the best way to elicit explanations from large models (compared to post-hoc explanation methods)?

Discussion Questions

- Would the inductive structure that was introduced in MolmoAct transpose well to other domains (locomotion, whole-body control, etc.)?
- In addition to depth perception and visual reasoning traces, what are other methods of chain of thought prompting to better ground VLAs?
- This paper proposes 2d drawings as an interface for steering VLAs? Is this the best way to steer VLAs?
- Is self-explainability the best way to elicit explanations from large models (compared to post-hoc explanation methods)?
- How do we evaluate the fidelity and faithfulness of the explanations generated by MolmoAct? Do we need a way to do this and if so, how could we go about it?

Thank You!

pick up the bowl yourself

deploy a robot with perception and state estimation pipelines to localize the bowl, plan a collision free path with inverse kinematics and trajectory optimization, follow the path with a stable controller to pick up the bowl

use underpaid grad students to collect data and 50k GPU hours to train a model that maps pixels to actions once every hour (pick up the bowl optional)

that's not a bowl.

References

[1] Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L., ... & Zhu, Y. (2025). Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734.

[2] Zawalski, M., Chen, W., Pertsch, K., Mees, O., Finn, C., & Levine, S. (2024). Robotic control via embodied chain-of-thought reasoning. arXiv preprint arXiv:2407.08693.

[3] Lee, J., Duan, J., Fang, H., Deng, Y., Liu, S., Li, B., ... & Krishna, R. (2025). Molmoact: Action reasoning models that can reason in space. arXiv preprint arXiv:2508.07917.

[4] Bigverdi, M., Luo, Z., Hsieh, C. Y., Shen, E., Chen, D., Shapiro, L. G., & Krishna, R. (2025). Perception tokens enhance visual reasoning in multimodal language models. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 3836-3845).

[5] Van Den Oord, A., & Vinyals, O. (2017). Neural discrete representation learning. Advances in neural information processing systems, 30.

[6] Zheng, R., Liang, Y., Huang, S., Gao, J., Daumé III, H., Kolobov, A., ... & Yang, J. (2024). Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. arXiv preprint arXiv:2412.10345.

[7] Deitke, M., Clark, C., Lee, S., Tripathi, R., Yang, Y., Park, J. S., ... & Kembhavi, A. (2025). Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 91-104).