VLMs are Native Action Models: Unleashing the manipulation capabilities of VLMs with Explicit Action-Space Grounding

Anonymous authors
Paper under double-blind review



Figure 1: **VLMs are native action models.** With explicit grounding in action spaces, pretrained VLMs can not only exhibit zero-shot manipulation capabilities without action pretraining, but also be further fine-tuned to improve existing skills and acquire capabilities for manipulation tasks beyond their zero-shot capabilities.

Abstract

Pretrained vision-language models (VLMs) already exhibit strong visual, spatial, and reasoning capabilities relevant to robotic manipulation. A key challenge in turning these capabilities into generalist manipulation policies is how to expose these capabilities through an appropriate action interface. Vision-language-action (VLA) models provide a unified trainable policy, but typically introduce learned action tokens, action heads, or action experts whose semantics must be acquired from robot trajectories before the model can act. In contrast, modular foundation-model-based manipulation methods can exploit pretrained VLM capabilities for zero-shot manipulation, but rely on external planning and control scaffolds that are not end-to-end trainable as a unified continuous-action policy. We introduce **EASG**, an ***Explicit Action-Space Grounding*** framework that bridges these two paradigms. EASG makes the semantics of a continuous robot action space directly observable in the VLM's perceptual context by projecting a tool center point (TCP)-centered action frame into first-person observations, while expressing actions through the VLM's native textual generation interface. This allows pretrained VLMs to directly translate their existing visual-spatial reasoning into executable robot actions, without robot-action pretraining or auxiliary learned modules. At the same time, the same interface supports direct fine-tuning with robot trajectories, allowing the model to continuously improve upon the manipulation capabilities unleashed by EASG. We demonstrate that EASG enables zero-shot manipulation and generalization across object positions, categories, and robot embodiments, while remaining effectively trainable in continuous action spaces. These results suggest that, given an appropriate action interface, pretrained VLMs can themselves serve as *native action models* for robotic manipulation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Advances in large-scale multimodal pretraining have equipped vision-language models (VLMs) with strong visual, spatial, and commonsense reasoning capabilities. Many of these capabilities are directly relevant to robotic manipulation. Yet turning a pretrained VLM into an executable manipulation policy requires an appropriate *action interface* that connects its perceptual and reasoning capabilities to robot control. Here, the *action space* specifies the control variables produced by the policy, such as incremental end-effector translation, rotation, and gripper commands. An action interface must make the semantics of this space, including its reference frame, parameterization, and physical meaning, accessible to the VLM, while also specifying how actions are represented to and generated by the model. Existing vision-language-action (VLA) models construct such an interface by extending pretrained VLMs with robot-action representations learned from robot trajectories. A common strategy treats robot actions as additional tokens (Zitkovich et al., 2023; Kim et al., 2025b; Pertsch et al., 2025; Qu et al., 2025), while other approaches introduce dedicated action modules, such as flow-matching or diffusion-based action experts (Black et al., 2024; NVIDIA et al., 2025) and continuous action heads trained with regression or parallel decoding objectives (Kim et al., 2025a). Alongside these architectural advances, VLA training has increasingly scaled to large and diverse robot datasets (O'Neill et al., 2024; Khazatsky et al., 2024; Wu et al., 2024; Bu et al., 2025). These approaches provide a unified and trainable action policy and have led to impressive advances in generalist robotic manipulation.

However, the semantics of these robot action spaces are generally not directly available to the pretrained VLM before robot-action training. Robot trajectories must therefore teach the model not only *what action to take*, but also how that action is defined in the physical world. First, the model must infer in which coordinate frame an action is expressed and how it is parameterized, since these conventions are typically only implicit in the action labels. Second, precise control requires sufficiently accurate metric geometry. The mapping from 2D visual observations to 3D physical motion is not uniquely determined without camera geometry(Hartley & Zisserman, 2004; Zhang et al., 2025; Jiang et al., 2025): variations in camera intrinsics and extrinsics can make visually similar observations correspond to different physical displacements in the robot. Third, the policy must learn how embodiment-specific robot states relate to end-effector motion in Cartesian space, including the relationship between robot configurations, end-effector states, and the resulting visual changes. Learning these geometric and kinematic correspondences jointly with manipulation behavior places a substantial burden on robot data and model capacity. More fundamentally, it prevents the manipulation-relevant capabilities already present in the pretrained VLM from being directly translated into executable task-space actions before this interface has been learned from robot trajectories. Extensive robot-action training may further alter some of the visual-semantic and reasoning capabilities inherited from the pretrained VLM (Driess et al., 2025; Hancock et al., 2026). An alternative line of work exposes pretrained foundation-model capabilities through explicit intermediate representations instead of learning the entire action interface from robot trajectories. **Modular foundation-model-based manipulation methods** generate policy programs (Liang et al., 2022), spatial constraints over object parts (Huang et al., 2024), composable 3D value maps (Huang et al., 2023), or relational keypoint constraints (Huang et al., 2025), which are subsequently converted into robot motion by downstream planners, optimizers, or controllers. By formulating manipulation problems in representations that pretrained foundation models can already reason about, these methods can perform a range of tasks zero-shot without robot-action pretraining. However, the mapping from visual reasoning to continuous robot actions is distributed across intermediate representations and external planning or execution modules, rather than represented as a unified action policy within the VLM itself. Consequently, the same VLM policy cannot be naturally improved end-to-end from robot trajectories through the interface that enables its zero-shot behavior.

The two paradigms therefore expose a complementary trade-off. VLA models provide a unified and trainable action policy, but require the action interface to be learned from robot data before the model can act. Modular foundation-model methods can directly exploit pretrained capabilities for zero-shot manipulation, but externalize action generation across separate representations, planners, and controllers. This raises a natural question: *Can an appropriate action interface enable a pretrained VLM itself to act as a unified manipulation policy, supporting zero-shot execution while continuing to improve through robot-data fine-tuning?*

To answer this question, we introduce **EASG**, an ***E**xplicit **A**ction-**S**pace **G**rounding* framework that provides a unified interface between pretrained VLM reasoning and continuous robot control. **EASG** explicitly grounds the semantics of a continuous robot action space in the VLM's perceptual context by projecting a tool center point (TCP)-centered action frame into first-person observations and providing metric information for action execution. Actions are expressed through the VLM's native textual generation interface, so the pretrained model can directly map its existing visual-spatial reasoning to continuous robot actions without first acquiring a new robot-specific output modality. Together, these designs allow pretrained VLMs to generate executable actions without robot-action pretraining or auxiliary learned planning modules. Crucially, the same action space and generation interface are preserved during robot-data fine-tuning, allowing the VLM to continuously improve the manipulation capabilities initially unleashed by **EASG**. To ensure reliable action generation through this native interface, we further introduce grammar-constrained decoding, which restricts autoregressive generation to valid action formats while preserving the pretrained VLM's original vocabulary and generation process. Because the pretrained VLM itself generates robot actions through the same native generation interface both before and after fine-tuning, we refer to it as a *native action model*. To summarize, our key contributions are as follows:

- We introduce **EASG**, an *Explicit Action-Space Grounding* framework that makes robot-action semantics directly accessible to pretrained VLMs while preserving their native textual generation interface, enabling them to act without large-scale robot-action pretraining.
- We show that the same action interface supports both zero-shot manipulation and robot-data fine-tuning.
- We empirically demonstrate the capability of VLM-native manipulation: a 27B-scale VLM with EASG achieves comparable zero-shot performance with fine-tuned VLA models, while a 0.8B-scale VLM fine-tuned with EASG achieves competitive performance with several VLA baselines.

2 Related Work

Vision-Language-Action (VLA) Models. Existing VLA models have explored discrete action tokens (Zitkovich et al., 2023; Kim et al., 2025b; Pertsch et al., 2025; Qu et al., 2025) or additional action modules (Black et al., 2024; NVIDIA et al., 2025; Kim et al., 2025a) for better robotic action representation and generation. With these architectural advances, VLA pretraining has scaled to increasingly large and diverse robot datasets (O'Neill et al., 2024; Khazatsky et al., 2024; Wu et al., 2024; Bu et al., 2025). More recently, video or world-model-based policies also try to leverage pretrained video generators to provide future visual dynamic priors for action generation (Kim et al., 2026; Ye et al., 2026). Despite these advances, the prevailing action-pretraining paradigm remains highly data-intensive, since it implicitly requires models to jointly infer action coordinates, depth information and inverse dynamics from robot data. We provide further discussion in Appendix A-1.

Modular Foundation-Model-Based Manipulation. Modular foundation-model-based manipulation methods leverage pretrained foundation models for high-level perception and reasoning, while mediating robot action through explicit intermediate representations and downstream planning or control modules. Code as Policies (Liang et al., 2022) generates executable programs that compose predefined perception and control APIs, while subsequent methods use structured representations such as 3D affordance maps in VoxPoser (Huang et al., 2023), spatial end-effector constraints in CoPa (Huang et al., 2024), and relational keypoint constraints in ReKep (Huang et al., 2025). These methods enable zero-shot manipulation without robot-action pretraining, and recent **Robot-Use-Agent** approaches further expose perception, planning, and control as callable tools or autonomously refine robot policies through execution feedback (Fu et al., 2026; Hu et al., 2026; Xiao et al., 2026; Lu et al., 2026). However, their perception-to-action mapping remains distributed across the foundation model and external execution modules rather than forming a unified action policy within the model itself, preventing end-to-end optimization of the complete perception-to-action pipeline. Further discussion is provided in Appendix A-2.

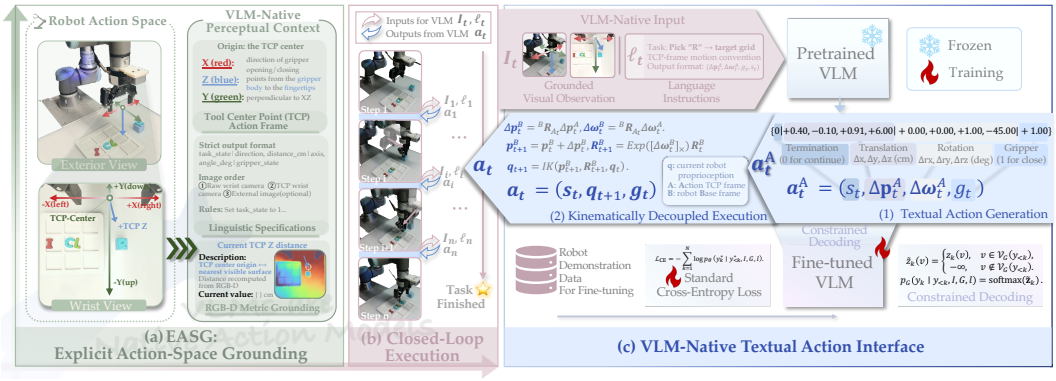

Figure 2: **Overview of EASG. EASG** explicitly grounds robot-action semantics in the VLM's perceptual context and expresses actions through its native textual generation interface, enabling both zero-shot manipulation and direct robot-data fine-tuning. Actions are generated in the TCP-centered frame(Action frame) and converted into low-level joint commands.

3 METHOD

Explicit Action-Space Grounding (EASG) makes the physical semantics of a robot action space directly observable in the VLM's perceptual context. We define the action-space grounding G_t as a structured representation that exposes the semantics of the continuous action space to the VLM:

$$G_t = (V_t, M_t, P_t), \tag{1}$$

where V_t denotes the visual projection of the TCP-centered action frame, M_t provides metric distance information by estimating the distance from the TCP center to the nearest visible surface along the TCP approach direction, and P_t specifies the linguistic semantics of the action parameterization and output format. At each control step t, the policy receives visual observations I_t, a language instruction l, and action-space grounding G_t, and predicts

$$a_t = \left(\Delta\mathbf{p}_t^A,\ \Delta\boldsymbol{\omega}_t^A,\ g_t,\ s_t\right), \tag{2}$$

where $\Delta\mathbf{p}_t^A \in \mathbb{R}^3$ is an incremental translation, $\Delta\boldsymbol{\omega}_t^A \in \mathbb{R}^3$ is an incremental axis–angle rotation vector, both expressed in the grounded action frame $\mathcal{F}_{A_t}$, g_t is the gripper command, and s_t indicates task completion. The action is executed, the environment is re-observed, and the process repeats in closed loop.

3.1 EXPLICIT ACTION-SPACE GROUNDING

EASG explicitly grounds the origin, axes, parameterization, and metric scale of the action space in the current observation, rather than requiring these semantics to be inferred implicitly from robot demonstrations. At each control step, we define a local action frame $\mathcal{F}_{A_t}$ whose origin is anchored at the current Tool Center Point (TCP). Its axes define the directions in which the VLM expresses incremental translation and rotation. Given calibrated camera geometry and the current robot state, the action frame is represented in the wrist-camera coordinate system as

$${}^{C}\mathbf{T}_{A_t} = \begin{bmatrix} {}^{C}\mathbf{R}_{A_t} & {}^{C}\mathbf{p}_{A_t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{3}$$

Let $\mathbf{K}$ denote the camera intrinsic matrix and $\pi(\cdot)$ the perspective projection operator. We project the action-frame origin and axes into the wrist-camera image:

$$\begin{aligned} \mathbf{u}_0 &= \pi\left(\mathbf{K}\,{}^{C}\mathbf{p}_{A_t}\right), \\ \mathbf{u}_j &= \pi\left[\mathbf{K}\left({}^{C}\mathbf{p}_{A_t} + \ell\,{}^{C}\mathbf{R}_{A_t}\mathbf{e}_j\right)\right], \qquad j \in \{x, y, z\}, \end{aligned} \tag{4}$$

where $\mathbf{e}_j$ is a canonical action direction and ℓ controls the visualized axis length. The accompanying prompt specifies axis directions, translation units, rotation parameterization, and gripper semantics. The VLM therefore both observes the action coordinate system in its visual context and receives

an explicit interpretation of its numerical outputs. To provide metric scale, we use aligned depth to estimate scene distance along the action-frame approach direction. For the ray

$$\mathbf{r}(d) = {}^{C}\mathbf{p}_{A_t} + d\,{}^{C}\mathbf{R}_{A_t}\mathbf{e}_z, \tag{5}$$

we estimate the distance d_t to the nearest visible surface from the RGB-D observation. This supplies metric 3D information directly, rather than requiring the VLM to infer precise physical distance from RGB appearance alone. The VLM predicts motion only in the grounded action frame $\mathcal{F}_{A_t}$; robot-specific coordinate transforms and kinematics are handled deterministically. Let $\mathcal{F}_B$ denote the robot base frame and ${}^{B}\mathbf{R}_{A_t}$ the orientation of the action frame expressed in $\mathcal{F}_B$. The predicted translation and rotation increments are first transformed into the robot base frame:

$$\Delta\mathbf{p}_t^B = {}^{B}\mathbf{R}_{A_t}\Delta\mathbf{p}_t^A, \qquad \Delta\boldsymbol{\omega}_t^B = {}^{B}\mathbf{R}_{A_t}\Delta\boldsymbol{\omega}_t^A. \tag{6}$$

Let $\mathbf{p}_t^B$ and $\mathbf{R}_t^B$ denote the current controlled end-effector position and orientation in the robot base frame. The corresponding target pose is

$$\mathbf{p}_t^{B,*} = \mathbf{p}_t^B + \Delta\mathbf{p}_t^B, \qquad \mathbf{R}_t^{B,*} = \mathrm{Exp}\left([\Delta\boldsymbol{\omega}_t^B]_\times\right)\mathbf{R}_t^B. \tag{7}$$

We denote the resulting target end-effector pose in the base frame by $\mathbf{T}_t^{B,*}$ and convert it into robot-specific commands through inverse kinematics and low-level control:

$$\mathbf{q}_t^* = \mathrm{IK}\left(\mathbf{T}_t^{B,*}; \mathbf{q}_t\right). \tag{8}$$

Thus, the VLM reasons in a common grounded action space, while deterministic geometry and embodiment-specific kinematics realize the predicted motion on each robot.

3.2 VLM-Native Textual Action Learning

We serialize continuous robot actions using only tokens from the pretrained VLM's native textual vocabulary, without introducing robot-specific action tokens or a separate continuous-action predictor. First, we parameterize translation by a unit direction $\mathbf{v}_t^A \in \mathbb{R}^3$ and a nonnegative distance d_t, and rotation by a unit axis $\mathbf{r}_t^A \in \mathbb{R}^3$ and a signed angle θ_t:

$$\Delta\mathbf{p}_t^A = d_t\,\mathbf{v}_t^A, \qquad \Delta\boldsymbol{\omega}_t^A = \theta_t\,\mathbf{r}_t^A, \tag{9}$$

where $\|\mathbf{v}_t^A\|_2 = 1$ for nonzero translation and $\|\mathbf{r}_t^A\|_2 = 1$ for nonzero rotation. Both directions are expressed in the grounded action frame $\mathcal{F}_{A_t}$. We represent d_t in centimeters and θ_t in degrees. In practice, the VLM may generate vectors that do not exactly satisfy the unit-norm constraint. We therefore project the generated vectors onto the unit sphere before executing the action. The resulting textual action follows

$$y_t = \texttt{ state } \texttt{|}\ v_x, v_y, v_z, d\ \texttt{|}\ r_x, r_y, r_z, \theta\ \texttt{|}\ \texttt{gripper }, \tag{10}$$

where every continuous scalar is emitted with an explicit + or − sign and exactly two decimal places. For example,

```
0|+0.40,-0.10,+0.91,+6.00| +0.00,+0.00,+1.00,-45.00|+1.00 .
```

The first field indicates task completion, while the final field specifies the desired gripper opening. Because all symbols already belong to the pretrained language vocabulary, the original autoregressive language decoder itself serves as the action decoder. The same textual action representation is used for both zero-shot execution and subsequent robot-data fine-tuning. Given a demonstrated action sequence $y^* = (y_1^*, \ldots, y_N^*)$, we therefore fine-tune the VLM directly with standard causal language-model cross-entropy:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{k=1}^{N} \log p_\theta\left(y_k^* \mid y_{<k}^*, I, G, l\right). \tag{11}$$

Fine-tuning improves the same autoregressive policy that already supports zero-shot manipulation, without changing its action interface. After fine-tuning, directly decoding actions as free-form language may produce invalid action formats, including malformed numeric values, missing fields, incorrect separators, or extraneous text. We therefore enforce a deterministic action grammar to

constrain decoding and ensure valid action generation. Let $\mathcal{V}_G(y_{<k})$ denote the set of tokens valid after prefix $y_{<k}$. For the original VLM logit $z_k(v)$, we use

$$\widetilde{z}_k(v) = \begin{cases} z_k(v), & v \in \mathcal{V}_G(y_{<k}), \\ -\infty, & \text{otherwise.} \end{cases} \tag{12}$$

Generation then proceeds autoregressively from $\mathrm{softmax}(\widetilde{\mathbf{z}}_k)$. The grammar constrains only command syntax—including field order, separators, signs, and numerical precision—while the numerical action itself remains predicted by the VLM. Further details are provided in Appendix B-2.

4 Experiments

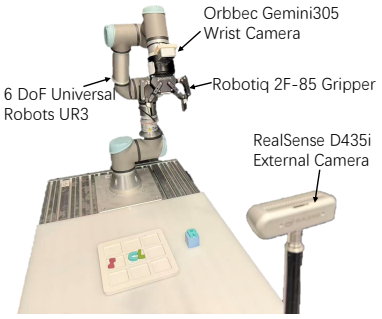

Figure 3: **Real-world robot setup.**

Experimental Setup. To study both the native zero-shot capability and the extensibility of VLM-based manipulation, we conduct real-world experiments on a 6-DoF UR3 robot with a wrist-mounted RGB-D camera and an external camera. We construct a task suite spanning rigid, articulated, deformable, precise, and long-horizon manipulation, and organize it into two groups of different difficulty. Task Group 1 contains relatively simple tasks, whereas Task Group 2 includes more challenging tasks requiring greater precision, more complex interactions, or longer-horizon execution. We find that pretrained VLMs with EASG exhibit non-trivial zero-shot capability on Task Group 1 but generally fail on Task Group 2. Accordingly, we use Task Group 1 to characterize native VLM manipulation and Task Group 2 to evaluate capability extension through robot-data fine-tuning.

Task Group 1: Zero-shot evaluation tasks.

Type	Task	Task description
Rigid	1.1.1	Grasp the cucumber
	1.1.2	Pick up the block
	1.1.3	Pick up the banana and put it on the plate
Articulated	1.2.1	Close the laptop
	1.2.2	Push the drawer closed
Deformable	1.3	Fold the towel

Task Group 2: Capability extension tasks.

Type	Task	Task description
Deformable	2.1	Fold the T-shirt in half.
Articulated	2.2	Open the laptop.
Precise	2.3.1	Pick up the ring and put it on one of the pegs.
	2.3.2	Hang the cup on the peg.
Long-horizon	2.4.1	Open the drawer, then put the Rubik's Cube in the drawer.
	2.4.2	Put the grapes in the drawer, then close the drawer.

Table 1: **Real-world task setups.** See Appendix C-1 for the full object group we included.

4.1 Zero-Shot VLM Manipulation via Action-Space Grounding

Q1. Can pretrained VLMs perform zero-shot manipulation with explicit action-space grounding?

Table 2: **Robot data efficiency comparison between EASG and fine-tuned VLAs.** Ours uses no task-specific demonstrations. VLA baselines are fine-tuned with 10/30/50 demonstrations per task.

Task	Ours	$\pi_{0.5}$			π_0-FAST			GR00T N1.6			OpenVLA-OFT		
	Zero	10	30	50	10	30	50	10	30	50	10	30	50
1.1.1	93.3	70.0	100.0	100.0	53.3	90.0	100.0	66.7	100.0	100.0	66.7	100.0	100.0
1.1.2	90.0	66.7	100.0	100.0	53.3	86.7	100.0	60.0	100.0	100.0	66.7	93.3	100.0
1.1.3	66.7	30.0	80.0	100.0	0.0	46.7	73.3	10.0	70.0	100.0	20.0	53.3	93.3
1.2.1	100.0	50.0	80.0	100.0	46.7	53.3	100.0	50.0	53.3	100.0	53.3	66.7	100.0
1.2.2	93.3	46.7	66.7	100.0	30.0	53.3	100.0	46.7	50.0	100.0	40.0	70.0	100.0
1.3	33.3	0.0	30.0	80.0	0.0	10.0	40.0	0.0	33.3	73.3	0.0	13.3	40.0
Avg.	79.4	**43.9**	**76.1**	**96.7**	30.6	56.7	85.6	38.9	67.8	95.6	41.1	66.1	88.9

Zero-shot Capability. To probe the native zero-shot manipulation capabilities of pretrained VLMs with EASG, we evaluate Qwen-3.8-27B on Task Group 1, as summarized in Table 1. For each task, we conduct three independent evaluation rounds with 10 trials per round, resulting in 30 trials in total. We report the mean success rate and standard deviation across the three evaluation rounds. The linguistic action-space grounding used in our experiments is provided in Appendix B-1.

Robot Data Needed to Match EASG Zero-Shot Performance. General-purpose VLAs typically require a small amount of task-specific data to adapt to a new embodiment and domain before executing newly specified tasks. We therefore fine-tune $\pi_{0.5}$ (Black et al., 2025), π_0-FAST (Pertsch et al., 2025), GR00T N1.6 (NVIDIA et al., 2025), and OpenVLA-OFT (Kim et al., 2025a) using 10/20/30/40/50 demonstrations per task. We find that approximately 30 demonstrations are sufficient for fine-tuned VLA policies to match the zero-shot success rate of EASG. Table 2 reports representative results at 10/30/50 demonstrations. The **full five-level comparison** is in Table 12.

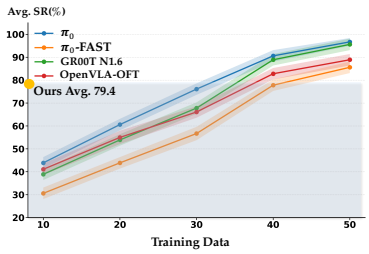

Figure 4: **Results visualization.**

Zero-Shot Generalization across Objects and Positions. Since EASG leverages the visual understanding acquired during VLM pretraining, we expect it to be naturally robust to variations in object positions, poses and color. We deploy three randomized configurations in Task Group 1, including pose, position and color. In Table 3, we find EASG largely preserves zero-shot performance of VLMs, whereas VLA baselines fine-tuned with 50 demonstrations exhibit a clear performance drop despite deliberate randomization of object poses and positions during VLA demonstration collection. For detailed setups and data analysis, see Appendix C-3.

Table 3: **Generalization under object and position randomization.** Randomized results are reported as mean ± Std. over three randomized configurations.

	Ours(Zero)	$\pi_{0.5}$(50)	π_0**-FAST**(50)	**GR00T N1.6**(50)	**OpenVLA-OFT**(50)
Randomized	77.3±5.9	40.2±8.7	25.5±9.5	37.8±10.6	28.1±10.9
Original	79.4±2.5	96.7±1.7	85.6±2.5	95.6±2.5	88.9±2.5

Findings. Explicit action-space grounding turns pretrained VLMs into zero-shot manipulation models with strong position, pose and color generalization.

4.2 Improving VLM-Native Manipulation through Post-Training

Q2. Can VLMs be fine-tuned beyond their zero-shot manipulation capabilities?

Fine-tuning VLMs in the Real World. We organize several more challenge tasks that cannot be completed by our EASG framework zero-shot , as depicted in Task Group 2, which contains manipulating deformable, articulated tasks, precise operation, and long-horizon tasks. We fine-tune the qwen-3.8-27B following the textual action learning procedure described in Sec. 3.2 with Lora (Hu et al., 2021). And we want to determine if it can extend its capability through finetuning; we also choose π_0-FAST, OpenVLA(Kim et al., 2025b), Octo(Ghosh et al., 2024) as baselines.

Table 4: **Results on harder real-world tasks. FT** denotes fine-tuning with teleoperated data. Gray cells indicate unavailable results, as pretrained VLA models generally need additional adaptation.

Method	**1.1.1**		**1.1.2**		**1.1.3**		**1.2.1**		**1.2.2**		**1.3**		**2.1**	**2.2**	**2.3.1**	**2.3.2**	**2.4.1**	**2.4.2**	**Avg.**
	Zero	*FT*	*Zero*	*FT*	*Zero*	*FT*	*Zero*	*FT*	*Zero*	*FT*	*Zero*	*FT*	*FT*	*FT*	*FT*	*FT*	*FT*	*FT*	*FT*
Ours	93.3	100.0	90.0	100.0	66.7	70.0	100.0	100.0	93.3	100.0	33.3	33.3	80.0	63.3	56.7	43.3	76.7	83.3	75.5
π_0-FAST	–	100.0	–	100.0	–	73.3	–	100.0	–	100.0	–	40.0	83.3	70.0	53.3	50.0	80.0	83.3	77.8
OpenVLA	–	100.0	–	100.0	–	60.0	–	100.0	–	100.0	–	23.3	70.0	46.7	43.3	20.0	73.3	76.7	67.8
Octo	–	100.0	–	100.0	–	63.3	–	100.0	–	100.0	–	30.0	73.3	40.0	46.7	26.7	80.0	76.7	69.7

Findings. VLMs can be fine-tuned beyond their zero-shot capabilities, matching fine-tuned large-scale pretrained VLAs on complex tasks in both simulation and the real world.

Fine-tuning VLMs in Simulation. To enable a controlled and reproducible comparison with pretrained VLAs, we select one widely used manipulation benchmark, Libero (Liu et al., 2023), to study the learning efficiency of fine-tuning VLMs. On LIBERO, Table 5 compares average success rates against VLAs with large-scale action pretraining. Unlike EASG's real-world training setup, we do not use depth information in simulation; the model receives **the same input observations** as the compared VLA baselines. We use Qwen3.5-0.8B as base model. For each task, we evaluate 16 rollouts per seed over three random seeds, resulting in 48 evaluation rollouts per task. Qwen3.5-0.8B surpass Octo and OpenVLA, with a close performance to π_0-FAST.

Method	Spatial	Object	Goal	10	Average
Ours(0.8B)	92.5±2.08	93.8±1.91	83.1±2.96	63.7±3.80	83.3±1.40
π_0-FAST	96.4	96.8	88.6	60.2	**85.5**
Octo	78.9	85.7	84.6	51.1	75.1
OpenVLA	84.7	88.4	79.2	53.7	76.5

Table 5: **Fine-tuning VLM against pretrained VLAs in simulation.**

4.3 Cross-Embodiment Transfer

Q3. Can VLM manipulation capabilities transfer across embodiments?

Our action-space grounding represents robot actions in the end-effector pose space rather than embodiment-specific joint coordinates. This embodiment-agnostic action-space grounding naturally raises the question of whether manipulation capabilities of a fine-tuned VLM can transfer across robot embodiments. We train the model using data collected by UR3 following the Sec. 3.2, with the exception that external-camera observations are not used in this experiment. We preserve scene configurations from Sec. 4.2 and replace the 6-DoF UR3 arm with a 7-DoF Flexiv robot, except external camera.We evaluate cross-embodiment transfer on Task Group 2 As shown in Table 6, the fine-tuned policies remain effective under the new embodiment, achieving success rates comparable to those on the original UR3 platform. For detailed cross-embodiment experimental settings, see Appendix E.

Task	UR3	Flexiv
2.1	76.7	73.3
2.2	60.0	56.7
2.3.1	53.3	50.0
2.3.2	36.7	40.0
2.4.1	76.7	73.3
2.4.2	80.0	80.0
Avg.	**63.9**	**62.2**

Table 6: **Cross-embodiment transfer.**

Findings. VLM-native manipulation capabilities can transfer across embodiments with EASG.

5 Capability Boundaries of VLM-Native Manipulation

5.1 Inference Settings for VLM-Native Manipulation

Q4. How do inference configurations extend the capability frontier of VLM-native manipulation?

We study two commonly available mechanisms: **memory** for interaction history across successive action predictions and **thinking** for additional reasoning. We evaluate their effects in a controlled 2× 2 factorial study on the same real-world Tasks 1.1–1.3 with Qwen-3.8-27B. This comparison allows us to distinguish how much performance gains arise from temporal context, additional reasoning and both.

Table 7: **Effect of memory and thinking on VLM-native zero-shot manipulation.**

	w/ Thinking							w/o Thinking						
Task	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.3	**Avg.**	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.3	**Avg.**
w/ Memory	93.3	90.0	66.7	100.0	93.3	33.3	79.4	13.3	16.7	0.0	16.7	23.3	0.0	11.7
w/o Memory	90.0	83.3	0.0	100.0	93.3	20.0	64.4	10.0	6.7	0.0	16.7	20.0	0.0	8.9

Findings. Thinking is a key factor for extending VLM-native manipulation capabilities, while memory provides complementary gains mainly for longer-horizon tasks through temporal context.

5.2 Zero-Shot Capability Frontier of VLMs

Q5. How do VLM family and scale shape the boundary of zero-shot manipulation?

We systematically characterize the **zero-shot manipulation frontier** of pretrained VLMs across model families and scales. We evaluate models ranging from Qwen-2B/27B/397B to larger frontier VLMs on Tasks 1.1–1.3 under the same EASG grounding and without any robot-action training. Rather than considering only average success rates, we examine **which tasks become solvable as model capability increases**. The task-wise scaling curves characterize how **model family, scale, and task complexity** jointly determine the boundary of VLM-native manipulation.We observe a clear scaling trend in VLM-native manipulation: task success rates consistently improve as model size increases from 2B to 27B parameters. However, further scaling from 27B to 397B yields limited gains, while the additional improvements achieved by the most advanced proprietary VLMs suggest that scaling alone may not fully determine manipulation capability.

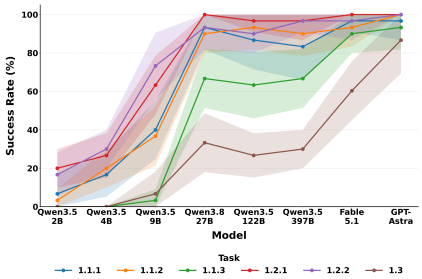

Figure 5: **Zero-shot capability frontier across VLM families and scales.**

6 Diagnostic Ablations

6.1 What Enables Effective Zero-Shot Action Grounding?

Action-Frame and Metric Grounding. We ablate **action-frame grounding** by replacing or removing the TCP-centered visual frame, and **metric-depth grounding** by fixing the projected z-axis length and removing explicit depth information. As shown in Table 8, removing either component drops the average success rate from 79.4% to 0%, indicating that both are critical for effective zero-shot manipulation. Further details are provided in Appendix F.

Table 8: **Effect of depth and frame grounding on real-world zero-shot manipulation.** The table reports performance on individual tasks under the same setup as experiments in Sec. 4.1.

	w/ Metric Depth							w/o Metric Depth						
Task	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.3	**Avg.**	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.3	**Avg.**
w/ Frame	93.3	90.0	66.7	100	93.3	33.3	79.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w/o Frame	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

6.2 Does GCD Improve Reliable Inference?

During post-training, VLMs must generate pose vectors through their native autoregressive decoding interface. Without additional constraints, small token-level generation errors can produce malformed or invalid action outputs. We introduce **Grammar Constrained Decoding (GCD)**, which restricts decoding to valid action formats . To isolate its effect, we fine-tune the Qwen3.5-0.8B with and without GCD and evaluate on Libero as the same setting in 4.2. As shown in Table 9, GCD consistently improves performance across all LIBERO suites. Removing GCD reduces the average success rate from 83.28% to 72.6% .

Method	**Spatial**	**Object**	**Goal**	**10**	**Average**
qwen3.5-0.8B	92.50±2.08	93.75±1.91	83.13±2.96	63.75±3.80	83.28±1.40
qwen3.5-0.8B (w.o. GCD)	81.25±3.09	78.75±3.23	72.50±3.53	58.13±3.90	72.66±1.73

Table 9: **Effect of grammar constrained decoding on simulation performance.**

AI USE STATEMENT

We used generative AI tools to assist with writing code and to improve the grammar, clarity, and readability of the manuscript. The authors reviewed and tested the AI-assisted code to verify its correctness and reviewed all AI-assisted text revisions to ensure that they accurately conveyed the intended scientific content. We take full responsibility for the final content of this work, including the code, experimental results, and manuscript.

ETHICS STATEMENT

This work investigates how pretrained vision-language models can serve as robotic manipulation policies through explicit action-space grounding. Such capabilities may support useful automation, but errors in visual perception, spatial reasoning, or action generation can result in unintended robot motions and physical damage. The proposed interface does not by itself provide safety guarantees, and grammar-constrained decoding ensures action-format validity rather than safe execution. Deployment beyond the evaluated settings therefore requires application-specific safety validation, appropriate operational constraints, and human oversight. The resulting policies may also inherit biases and failure modes from their pretrained backbones, which should be considered when assessing their suitability for new tasks and environments.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 3 specifies the action-space grounding, native textual action representation, coordinate transformations for execution, and fine-tuning objective. Appendix B-1 provides the complete zero-shot prompts, including linguistic action-space grounding, observation structure, action semantics, and output-format requirements. Appendix B-2 details the action grammar, finite-state prefix validation, token-level logit masking, and implementation of grammar-constrained decoding. Further experimental details are provided in the appendix sections on zero-shot manipulation and fine-tuning evaluation, together with additional ablation studies. We also provide an anonymous GitHub repository containing supplementary implementation details and resources to further support reproducibility: https://anonymous.4open.science/r/anonymous-code-7B20/..

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the Conference on Robot Learning*, pp. 17–40, 2025.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3549–3556. IEEE, 2025. doi: 10.1109/IROS60139.2025.11247088.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113969–113990, 2025. doi: 10.52202/085713-3439.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Jim Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation, 2026. URL https://arxiv.org/abs/2603.22435.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

Asher J. Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *The Fourteenth International Conference on Learning Representations*, 2026.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hengyuan Hu, Priya Sundaresan, Jensen Gao, and Dorsa Sadigh. VIA: Visual interface agent for robot control, 2026. URL https://arxiv.org/abs/2607.11119.

Haoxu Huang, Fanqi Lin, Yingdong Hu, Shengjie Wang, and Yang Gao. Copa: General robotic manipulation through spatial constraints of parts with foundation models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9488–9495, 2024. doi: 10.1109/IROS58592.2024.10801352.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pp. 4573–4602, 2025.

Tianchong Jiang, Jingtian Ji, Xiangshan Tan, Jiading Fang, Anand Bhattad, Vitor Guizilini, and Matthew R. Walter. Do you know where your camera is? view-invariant policy learning with camera conditioning, 2025. URL https://arxiv.org/abs/2510.02268.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning*, 2025b.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Jim Fan, and Guanzhi Wang. ASPIRE: Agentic /skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026. URL https://arxiv.org/abs/2607.00272.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, Jiayuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Shichao Fan, Xinhua Wang, Fei Liao, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Ning Liu, Pei Ren, Qiang Zhang, Yaoxu Lyu, Mengzhen Liu, Jingyang He, Yulin Luo, Zeyu Gao, Chenxuan Li, Chenyang Gu, Yankai Fu, Di Wu, Xingyu Wang, Sixiang Chen, Zhenyu Wang, Pengju An, Siyuan Qian, Shanghang Zhang, and Jian Tang. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Max Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Jim Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic robot policy self-improvement in the real world, 2026. URL https://arxiv.org/abs/2606.19980.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Tianyi Zhang, Haonan Duan, Haoran Hao, Yu Qiao, Jifeng Dai, and Zhi Hou. Grounding actions in camera space: Observation-centric vision-language-action policy, 2025. URL https://arxiv.org/abs/2508.13103.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

APPENDIX

A Full Related Work Analysis 15

A-1 Vision-Language-Action Models 15

A-2 Code As Policies and Robot-Use Agents 15

B EASG Method Details 17

B-1 Zero-shot Action-Space Grounding Prompts 17

B-2 Implementation of Grammar-Constrained Decoding 20

C Zero-shot Experiments 21

C-1 Experiment Settings 21

C-2 Baseline Models 21

C-3 Randomization Settings 23

D Fine-tuning Protocols of VLMs with EASG 24

D-1 Tasks and Metrics of Simulation Benchmarks 24

D-2 Tasks and Metrics of Real-world Scenes 24

D-3 Real-world Training Protocol 25

E Cross-Embodiment 25

E-1 Experiment Design 25

F Full Analysis of Ablation Study 27

F-1 What Enables Effective Zero-Shot Action-Space Grounding? 27

A Full Related Work Analysis

A-1 Vision-Language-Action Models

Vision-Language-Action (VLA) models extend pretrained vision-language models (VLMs) with robot action generation and have emerged as a promising paradigm for general-purpose robotic manipulation. One family of approaches preserves the autoregressive next-token prediction interface of VLMs by **representing robot actions as discrete tokens**. RT-2 and OpenVLA discretize each scalar action dimension into 256 bins and map the resulting bin indices to tokens in the language-model vocabulary, such that a continuous robot action is generated autoregressively as a sequence of action tokens (Zitkovich et al., 2023; Kim et al., 2025b). SpatialVLA similarly predicts discrete spatial action tokens, but constructs adaptive action grids according to the empirical action distribution to better encode translation and rotation in 3D space (Qu et al., 2025). FAST instead operates on action chunks: it transforms normalized action sequences into the frequency domain using a discrete cosine transform (DCT), quantizes and flattens the resulting coefficients, and further compresses them with byte-pair encoding (BPE) into compact action-token sequences (Pertsch et al., 2025).

Another family directly augments the VLM with **dedicated continuous-action modules**. π_0 attaches an action expert that generates continuous action chunks through conditional flow matching (Black et al., 2024), while GR00T N1 couples its vision-language module with a diffusion-transformer action module (NVIDIA et al., 2025). OpenVLA-OFT instead replaces discrete token prediction with an MLP action head that maps VLM hidden states directly to continuous actions, using parallel decoding, action chunking, and an ℓ_1 regression objective (Kim et al., 2025a). Together with these architectural advances, VLA pretraining has scaled to increasingly large and diverse robot datasets (O'Neill et al., 2024; Khazatsky et al., 2024; Wu et al., 2024; Bu et al., 2025). This paradigm integrates semantic perception, language understanding, and action prediction within a single trainable model, enabling manipulation capabilities to improve with increasing robot data and supporting multi-task manipulation, language-conditioned control, and generalization across objects and environments.

Despite this progress, the prevailing action-pretraining paradigm leaves several fundamental grounding problems to be learned implicitly from robot trajectories. First, the model must infer the **semantics of the action coordinate frame**. Cartesian translations and rotations are commonly supervised in robot-centric frames such as the robot base frame, while the relationship between this action frame and the visual observation is rarely made explicit. Consequently, the model must learn simultaneously *what motion should be executed* and *how that motion is represented in the coordinate system of a particular robot*. Second, the model must **recover sufficiently accurate depth information** from RGB observations for precise distance control.

Finally, action generation requires learning **kinematic, visual-geometric, and inverse-dynamics mappings**, including mappings from joint configurations to end-effector poses and from RGB observations to executable poses. These mappings are themselves nontrivial even when treated as isolated problems, yet conventional VLA training must learn them jointly and implicitly together with task-level manipulation behavior.

As a result, substantial amounts of robot data are spent not only on learning manipulation behaviors, but also on recovering geometric relationships among observations, robot embodiments, and action spaces, leading to substantial data and training costs. Moreover, the distribution mismatch between multimodal pretraining and robot action data (Driess et al., 2025; Hancock et al., 2026) may further degrade pretrained reasoning capabilities.

A-2 Code As Policies and Robot-Use Agents

There exists a group of prior works that directly exploit the native reasoning capabilities of pretrained foundation models for generalist manipulation without large-scale action pretraining. These methods generally follow a *code-as-policy* paradigm, where foundation models construct explicit *intermediate representations* that are subsequently converted into executable robot actions through external perception, planning, or control modules.

Code-as-Policy Methods. Code as Policies (Liang et al., 2022) formulates robot policies as executable programs generated by code-pretrained language models. The generated programs process perception outputs and compose predefined control APIs, while hierarchical code generation recursively synthesizes auxiliary functions required by more complex instructions. VoxPoser (Huang et al., 2023) replaces explicit policy programs with composable 3D affordance and constraint value maps. Language models specify spatial objectives and constraints, which are grounded into 3D observations with visual models and subsequently optimized by a model-based motion planner to produce dense 6-DoF end-effector trajectories. CoPa (Huang et al., 2024) instead grounds task-relevant object parts, identifies their spatial constraints with foundation models, and derives a sequence of task-aware 6-DoF end-effector poses that can be integrated with standard robot planning algorithms. ReKep (Huang et al., 2025) represents manipulation objectives as *relational keypoint constraints*: Python functions map 3D keypoint configurations to scalar costs, and a hierarchical constrained optimizer solves these costs to generate closed-loop SE(3) end-effector trajectories.

These methods therefore expose different intermediate interfaces between foundation-model reasoning and physical control—programs, 3D value maps, spatial constraints, or relational keypoint constraints—while relying on downstream planners, optimizers, and robot controllers for action execution.

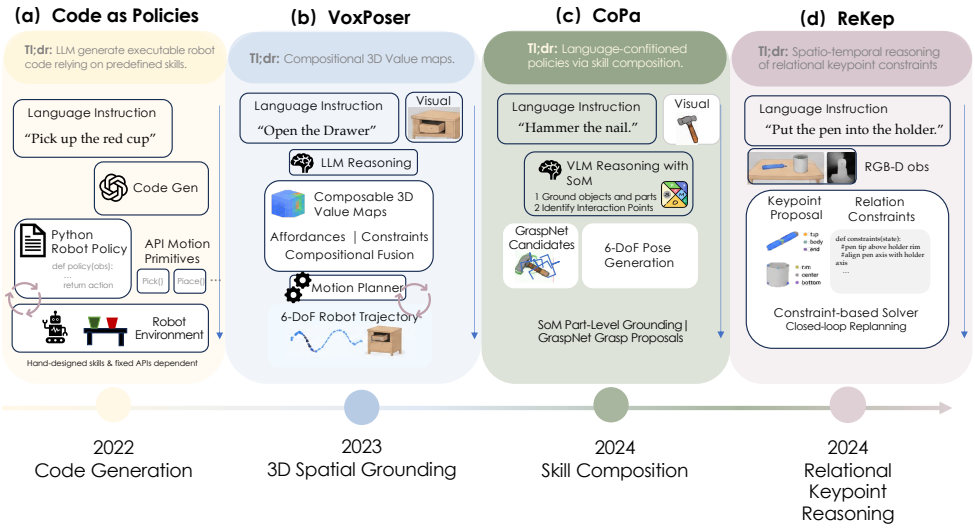

Figure 6: **Comparison with the most related methods.** We contrast EASG with representative prior methods to highlight the core differences in action representation, execution interface, action pretraining, zero-shot manipulation, and policy fine-tuning.

Robot-Use Agents. Recent progress in agentic foundation models further generalizes this paradigm toward systems that we broadly refer to as *Robot-Use Agents*. Instead of relying on a single fixed intermediate representation, these systems expose perception, planning, execution, and robot-control modules as callable tools that can be orchestrated by a general-purpose agent. CaP-X (Fu et al., 2026), for example, provides coding agents with perception and control primitives and systematically studies different levels of abstraction, interaction, and visual grounding. Its agentic variants additionally exploit multi-turn execution feedback, visual differencing, and automatic skill synthesis, while CaP-RL post-trains the coding model with verifiable task rewards to generate more effective robot programs. VIA (Hu et al., 2026) takes a different approach by presenting an RGB-D-reconstructed 3D visual interface to an off-the-shelf coding agent. The agent observes screenshots, manipulates a virtual target gripper through a small set of callable tools, and delegates execution of the resulting waypoints to a low-level robot controller.

Recent systems further explore autonomous policy improvement through execution feedback. EMPIRE (Xiao et al., 2026) treats a coding agent as an autonomous policy developer that can collect rollouts, diagnose failures, modify policy or training code, and invoke different policy-learning strategies through an automated robot experimentation infrastructure. ASPIRE (Lu et al., 2026)

instead iteratively diagnoses failures from multimodal execution traces, synthesizes and validates program repairs, and accumulates successful repairs into an evolving reusable skill library. Importantly, these approaches introduce meaningful forms of *system-level* improvement, but the learned object remains the generated program, skill scaffold, coding agent, or a separate downstream policy rather than the pretrained foundation model acting directly as a continuous-action predictor.

Limitations of Modular Action Interfaces. Such systems demonstrate that pretrained foundation models can accomplish a subset of manipulation tasks zero-shot without conventional large-scale robot-action pretraining. However, their capabilities are mediated by the surrounding tool pipeline. The foundation model does not **directly generate executable continuous robot actions**; instead, its outputs must be interpreted through **manually or automatically constructed programs**, geometric representations, motion planners, trajectory optimizers, or semantic control tools. Consequently, performance can depend strongly on the abstractions and capabilities exposed by the external scaffold.

This modular design can additionally introduce considerable ***system complexity***. Execution may require multiple perception models, geometric processing stages, planning or optimization modules, code execution environments, and robot-specific control interfaces. These dependencies can increase planning latency and engineering overhead, particularly for closed-loop manipulation requiring frequent visual feedback and action updates.

More importantly, when native zero-shot reasoning becomes insufficient, these modular pipelines lack a unified mechanism for ***fine-tuning the foundation models in continuous action spaces***. Classical code-as-policy methods largely retain fixed foundation models and improve execution through their external representations and planners (Liang et al., 2022; Huang et al., 2023; 2024; 2025). More recent agentic systems can improve generated programs, learn reusable skills, post-train coding agents, or train separate downstream policies (Fu et al., 2026; Xiao et al., 2026; Lu et al., 2026); however, these forms of adaptation do not directly turn the same pretrained foundation model used for zero-shot reasoning into a continuous robot-action policy.

We make a short summary of the above discussion and comparison in Table 10.

B EASG Method Details

B-1 Zero-shot Action-Space Grounding Prompts

We provide the complete prompts used for zero-shot action-space grounding in this section. These prompts specify the grounded action representation, the interpretation of the projected TCP-centered coordinate frame, and the required output format for direct robot pose generation. Unless otherwise stated, the same prompt structure is used across zero-shot experiments, with only task-specific instructions and observations changed accordingly.

Linguistic Action-Space Grounding

Control a robotic arm using the current visual observations. The wrist camera is mounted near a parallel-jaw gripper, and the processed image marks the TCP control point and coordinate axes; these graphics are not objects to grasp.
Express all motion commands in the TCP coordinate frame attached to the gripper. Its origin is the TCP center:

- X (red): runs across the gap between the two fingers, along the gripper's opening and closing direction.
- $+Z$ (blue): points forward along the fingers, from the gripper body toward the fingertips.
- Y (green): is perpendicular to the plane formed by X and Z.

The colored arrows indicate the positive directions. The axes form a right-handed coordinate frame:

$$+X \times +Y = +Z.$$

Negative directions are opposite to the corresponding positive directions.

Table 10: **Comparison with representative code-as-policy and Robot-Use Agent methods.** Prior approaches connect foundation-model reasoning to physical control through explicit intermediate representations or external tools, whereas EASG directly grounds continuous robot poses into the VLM native interface.

Method	Action Interface	Execution Pipeline	Further Improvement
Code as Policies (Liang et al., 2022)	Executable policy programs	• Perception APIs • Geometric operations • Robot-control APIs	No foundation-model training
VoxPoser (Huang et al., 2023)	3D affordance and constraint maps	• RGB-D grounding • 3D value-map construction • Model-based motion planning	No foundation-model training
CoPa (Huang et al., 2024)	Part-level spatial constraints and EEF poses	• Object-part grounding • Grasp proposal • Geometric constraint solving	No foundation-model training
ReKep (Huang et al., 2025)	Relational keypoint constraints	• Keypoint perception • Constraint generation • Hierarchical SE(3) optimization	No foundation-model training
CaP-X (Fu et al., 2026)	Programs over perception/control primitives	• Perception/control tools • Program synthesis • Execution feedback	Skills & coding-agent RL
VIA (Hu et al., 2026)	Visual tool calls and target-gripper waypoints	• RGB-D 3D interface • Target-gripper control • Low-level execution	No foundation-model training
ENPIRE (Xiao et al., 2026)	Policy and training code	• Robot rollouts • Failure diagnosis • Downstream policy training	Policy learning; no FM training
ASPIRE (Lu et al., 2026)	Policy programs and reusable skills	• Execution tracing • Program repair • Skill validation	Skills & iterative refinement
EASG (Ours)	**Continuous pose vectors**	• **Camera-projected grounding** • **TCP-centered representation** • **Direct VLM action prediction**	**Continuous-action space VLM fine-tuning**

These arrows are projections of the gripper's 3D axes onto the image, not the camera's image axes. The axes are perpendicular in 3D, but their projections may appear tilted or non-perpendicular.
Output one incremental action, expressing both the translation direction and rotation axis in the TCP frame before the action is executed. Translation specifies the displacement of the TCP center; rotation specifies the orientation change about that center as a unit axis and a signed angle.
Image order when multiple images are provided:

1. Raw wrist-camera image.
2. Processed wrist-camera image with the TCP-center coordinate frame.
3. External overview image, if provided: use it to understand the global scene, object identity, occlusions, and task completion, not to define motion axes.

Current TCP Z distance information:
`[TCP_Z_DISTANCE_INFO]`
The blue endpoint marks the visible surface point used to estimate clearance along TCP $+Z$, not a fixed axis length. The estimate may refer to a surface other than the target and does not guarantee clearance for the entire gripper. Use it together with finger-object alignment to judge grasping range.
Current gripper state:
`[Gripper state]`
Gripper opening ranges continuously from 0 (fully closed) to 1 (fully open). The input is the measured opening; the output `gripper_state` is the desired opening, not a change in opening.
Task:
`[Task]`
Return one line in the exact format below, without spaces, unit suffixes, or extra text. `task_state` must be the integer 0 or 1; all other values must be finite and include an explicit $+$ or $-$ sign and exactly two decimal places.
Strict output format:

```
task_state|direction_x,direction_y,direction_z,distance_cm|
axis_x,axis_y,axis_z,angle_deg|gripper_state
```

Rules:

- Set `task_state` to 1 only when current observations confirm completion of the entire task; otherwise use 0.
- If `task_state` is 1, output a no-op: zero translation, zero rotation, and an unchanged gripper state.
- For nonzero translation or rotation, normalize the direction or axis to a 3D unit vector before rounding its components.
- `distance_cm` is the nonnegative translation distance in centimeters.
- If no translation is needed, use

  `+0.00,+0.00,+0.00,+0.00`

  for the translation field.
- `angle_deg` is the signed rotation angle in degrees, following the right-hand rule around the specified axis.
- If no rotation is needed, use

  `+0.00,+0.00,+1.00,+0.00`

  for the rotation field.

Example:

```
0|+0.40,-0.10,+0.91,+6.00|+0.00,+0.00,+1.00,-45.00|+1.00
```

B-2 Implementation of Grammar-Constrained Decoding

Although the native language decoder allows the VLM to express continuous robot actions without introducing a specialized action vocabulary, unrestricted generation may produce malformed commands or additional natural-language text. We therefore apply grammar-constrained autoregressive decoding at inference time. The constraint operates only on the syntactic form of the action and does not alter the relative logits of valid tokens.

Action grammar. For the single-arm policy, each generated action has the form

```
0|+0.40,-0.10,+0.91,+6.00| +0.00,+0.00,+1.00,-45.00|+1.00 .
```

The four fields correspond to task state, translation, rotation, and gripper state, respectively. The translation field contains a 3D unit direction and a distance in centimeters, while the rotation field contains a 3D unit axis and a signed angle in degrees.

We define the valid action language using the following grammar:

$$
\begin{aligned}
\langle \text{action} \rangle &::= \langle \text{state} \rangle \mid \langle \text{trans} \rangle \mid \langle \text{rot} \rangle \mid \langle \text{grip} \rangle, \\
\langle \text{state} \rangle &::= \texttt{0} \mid \texttt{1}, \\
\langle \text{trans} \rangle &::= \langle n \rangle \texttt{,} \langle n \rangle \texttt{,} \langle n \rangle \texttt{,} \langle n \rangle, \\
\langle \text{rot} \rangle &::= \langle n \rangle \texttt{,} \langle n \rangle \texttt{,} \langle n \rangle \texttt{,} \langle n \rangle, \\
\langle \text{grip} \rangle &::= \langle n \rangle, \\
\langle n \rangle &::= \langle \text{sign} \rangle \langle \text{int} \rangle \texttt{.} \langle \text{digit} \rangle \langle \text{digit} \rangle, \\
\langle \text{sign} \rangle &::= \texttt{+} \mid \texttt{-}, \qquad \langle \text{int} \rangle ::= \langle \text{digit} \rangle^{+}, \\
\langle \text{digit} \rangle &::= \texttt{0} \mid \cdots \mid \texttt{9}.
\end{aligned}
\tag{13}
$$

Thus, every continuous quantity must contain an explicit sign and exactly two digits after the decimal point. The grammar enforces only serialization syntax. Semantic constraints—such as unit-norm translation directions and rotation axes, nonnegative translation distance, valid gripper range, and task-completion semantics—are specified by the policy prompt and handled independently of the decoding grammar.

Finite-state prefix validation. We compile the grammar into a deterministic finite-state machine. Let q_k denote its state after the currently generated textual prefix $y_{<k}$, and let

$$\delta(q, c)$$

denote the transition produced by consuming character c. An invalid transition enters a rejecting state $\perp$. For a character string $s = (c_1, \ldots, c_m)$, we write

$$\delta^*(q, s) = \delta(\cdots \delta(\delta(q, c_1), c_2), \ldots, c_m) \tag{14}$$

for the state obtained after consuming the complete string.

Importantly, the constraint is applied at the *token* level rather than assuming a one-to-one correspondence between tokens and characters. Let $\tau(v)$ denote the text emitted by vocabulary token v under the pretrained tokenizer. A token is considered valid after prefix $y_{<k}$ iff its entire decoded string can be consumed by the grammar:

$$\mathcal{V}_G(q_k) = \{ v \in \mathcal{V} \mid \delta^*(q_k, \tau(v)) \neq \perp \}. \tag{15}$$

This treatment naturally supports subword tokens that span multiple grammar symbols, such as a sign followed by digits or a delimiter followed by the beginning of the next field.

The end-of-sequence token is permitted only when the grammar is in an accepting state, i.e., after the second decimal digit of the final gripper value. Whitespace, natural-language tokens, malformed separators, missing signs, and numbers with an invalid decimal format are therefore rejected before they can be generated.

Logit masking. At autoregressive step k, the VLM first produces its original vocabulary logits $z_k(v)$. We preserve these logits for grammar-valid tokens and mask all others:

$$\widetilde{z}_k(v) = \begin{cases} z_k(v), & v \in \mathcal{V}_G(q_k), \\ -\infty, & \text{otherwise.} \end{cases} \tag{16}$$

The next token is then selected from $\mathrm{softmax}(\widetilde{\mathbf{z}}_k)$ using the same decoding rule as the underlying VLM. After token y_k is selected, the grammar state is updated as

$$q_{k+1} = \delta^* \left(q_k, \tau(y_k)\right). \tag{17}$$

Consequently, constrained decoding changes only the support of the next-token distribution: among all syntactically valid continuations, token preferences remain those of the original pretrained language decoder.

Implementation. We implement the grammar as an inference-time logits processor. Generation starts from the grammar's initial state at the first assistant-output token. At every decoding step, the processor determines the vocabulary tokens whose decoded strings admit a valid transition from the current grammar state, masks all remaining logits, and advances the state using the selected token. Because the grammar contains no learned parameters and uses only the pretrained tokenizer vocabulary, it introduces neither an additional action head nor robot-specific action tokens.

C Zero-shot Experiments

Zero-shot Experiments. In our setting, zero-shot execution refers to directly deploying a pretrained VLM to perform a manipulation task without any robot-action training, using the explicit action-space grounding (**EASG**) described in Sec. 3 and further detailed in Appendix B. At each control step, the VLM receives the current visual observations, the task instruction, and explicit action-space grounding that combines visual, metric, and linguistic cues. It then autoregressively outputs a robot action represented entirely in plain text. The textual action specifies task completion, incremental translation, incremental rotation, and gripper control in the grounded action frame. The predicted action is executed, after which the environment is re-observed and the process repeats in closed loop.

C-1 Experiment Settings

Real-World Hardware Setup. Our real-world experiment platform consists of a Universal Robots UR3 arm with a Robotiq 2F-85 gripper, operating over a 60×60 cm tabletop workspace. We use a wrist-mounted Orbbec Gemibi 305 camera and a fixed external Intel RealSense D435i camera to provide visual observations. Unless otherwise stated (for example, in the experiment of cross-embodiment in Sec. 4.3), all real-world experiments share the same robot, camera configuration, and workspace, both the tasks tested in the zero-shot experiments and the tasks tested in the fine-tuning experiments.

Task Composition Analysis. We construct a diverse set of real-world manipulation tasks covering rigid-object manipulation, deformable-object manipulation, articulated-object interaction, precise/contact-rich manipulation, and long-horizon multi-stage tasks. The task definitions and corresponding success criteria are summarized in Table 11.

Representative Zero-shot Inference Trajectories. For each task, we select one successful zero-shot trajectory according to the task-specific success metrics defined in Table 11. Figure 8 visualizes these representative trajectories.

C-2 Baseline Models

VLA Baselines. We compare EASG against $\pi_{0.5}$, π_0, π_0-FAST, and OpenVLA-OFT, representing recent general-purpose VLA policies with different model architectures and action-generation schemes. Each baseline is fine-tuned on task-specific demonstrations collected with the same embodiment, workspace, and task definitions used for evaluating EASG. As presented in Table 12, we



Figure 7: **Objects used in our real-world manipulation experiments.**

Table 11: **Task Group 1: Zero-shot evaluation tasks (Easy).**

Type	Task	Task description	Success criterion
Rigid	1.1.1	Grasp the cucumber	The cucumber is successfully grasped and held by the robot.
	1.1.2	Pick up the block	The block is lifted off the table and held by the robot.
	1.1.3	Pick up the banana and put it on the plate	The banana is lifted from the table and placed on the plate.
Articulated	1.2.1	Close the laptop	The laptop lid reaches the closed position.
	1.2.2	Push the drawer closed	The drawer is fully closed.
Deformable	1.3	Fold the towel	The towel is folded with partial overlap between folded regions.

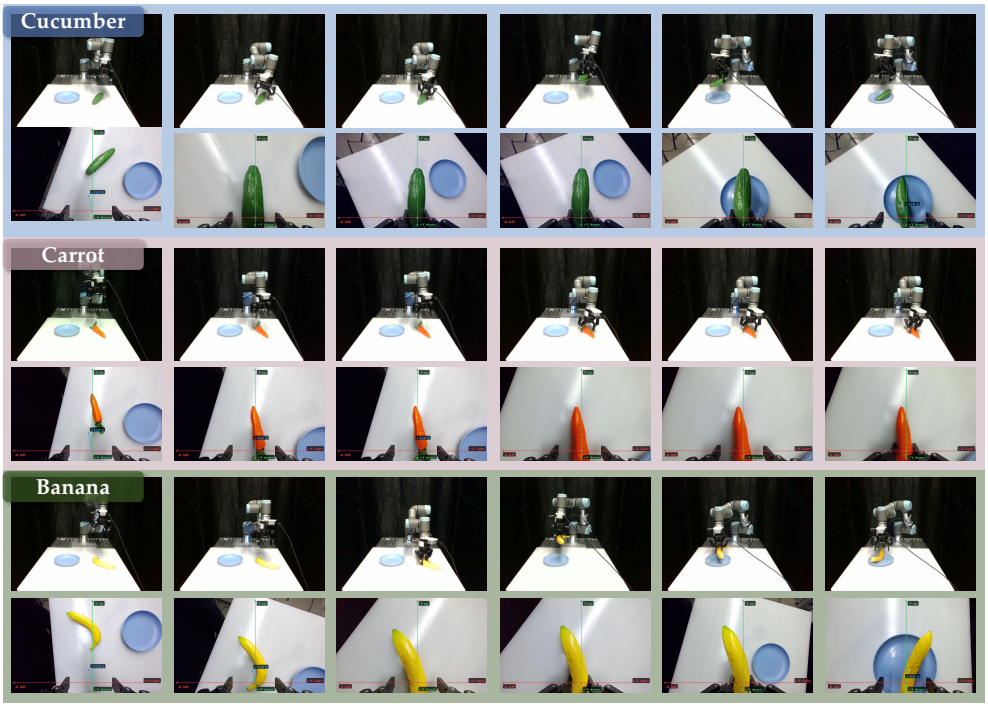

Figure 8: **Demonstration of successful zero-shot rollout cases.**

vary the amount of fine-tuning data from 10 to 50 demonstrations per task to measure how much robot data is required for these pretrained VLAs to match the zero-shot performance of EASG. Detailed results across all data budgets are reported below.

Table 12: **Full comparison with different amounts of robot data.** We report mean success rates ± standard deviations for VLA baselines fine-tuned with 10/20/30/40/50 demonstrations per task. Ours uses no task-specific demonstrations.

Method	Demos.	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.3	Avg.
Ours	*Zero*	93.3 ± 5.8	90.0 ± 10.0	66.7 ± 15.3	100.0 ± 0.0	93.3 ± 5.8	33.3 ± 15.3	79.4 ± 2.5
$\pi_{0.5}$	10	70.0 ± 10.0	66.7 ± 15.3	30.0 ± 10.0	50.0 ± 10.0	46.7 ± 15.3	0.0 ± 0.0	43.9 ± 2.5
	20	86.7 ± 15.3	83.3 ± 15.3	56.7 ± 15.3	66.7 ± 15.3	56.7 ± 15.3	13.3 ± 15.3	60.6 ± 2.5
	30	100.0 ± 0.0	100.0 ± 0.0	80.0 ± 10.0	80.0 ± 10.0	66.7 ± 15.3	30.0 ± 10.0	76.1 ± 2.5
	40	100.0 ± 0.0	100.0 ± 0.0	93.3 ± 5.8	93.3 ± 5.8	90.0 ± 10.0	66.7 ± 15.3	90.6 ± 2.5
	50	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	80.0 ± 10.0	96.7 ± 1.7
π_0-FAST	10	53.3 ± 15.3	53.3 ± 15.3	0.0 ± 0.0	46.7 ± 15.3	30.0 ± 10.0	0.0 ± 0.0	30.6 ± 2.5
	20	73.3 ± 15.3	70.0 ± 10.0	23.3 ± 15.3	50.0 ± 10.0	43.3 ± 15.3	3.3 ± 5.8	43.9 ± 2.5
	30	90.0 ± 10.0	86.7 ± 15.3	46.7 ± 15.3	53.3 ± 15.3	53.3 ± 15.3	10.0 ± 10.0	56.7 ± 2.9
	40	96.7 ± 5.8	96.7 ± 5.8	66.7 ± 15.3	86.7 ± 15.3	90.0 ± 10.0	30.0 ± 10.0	77.8 ± 2.5
	50	100.0 ± 0.0	100.0 ± 0.0	73.3 ± 15.3	100.0 ± 0.0	100.0 ± 0.0	40.0 ± 10.0	85.6 ± 2.5
GR00T N1.6	10	66.7 ± 15.3	60.0 ± 10.0	10.0 ± 10.0	50.0 ± 10.0	46.7 ± 15.3	0.0 ± 0.0	38.9 ± 2.5
	20	86.7 ± 15.3	83.3 ± 15.3	36.7 ± 15.3	53.3 ± 15.3	50.0 ± 10.0	13.3 ± 15.3	53.9 ± 2.5
	30	100.0 ± 0.0	100.0 ± 0.0	70.0 ± 10.0	53.3 ± 15.3	50.0 ± 10.0	33.3 ± 15.3	67.8 ± 2.5
	40	100.0 ± 0.0	100.0 ± 0.0	93.3 ± 5.8	90.0 ± 10.0	90.0 ± 10.0	60.0 ± 10.0	88.9 ± 2.5
	50	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	73.3 ± 15.3	95.6 ± 2.5
OpenVLA-OFT	10	66.7 ± 15.3	66.7 ± 15.3	20.0 ± 10.0	53.3 ± 15.3	40.0 ± 10.0	0.0 ± 0.0	41.1 ± 2.5
	20	86.7 ± 15.3	83.3 ± 15.3	36.7 ± 15.3	60.0 ± 10.0	56.7 ± 15.3	6.7 ± 5.8	55.0 ± 2.9
	30	100.0 ± 0.0	93.3 ± 5.8	53.3 ± 15.3	66.7 ± 15.3	70.0 ± 10.0	13.3 ± 15.3	66.1 ± 2.5
	40	100.0 ± 0.0	100.0 ± 0.0	83.3 ± 15.3	90.0 ± 10.0	93.3 ± 5.8	30.0 ± 10.0	82.8 ± 2.5
	50	100.0 ± 0.0	100.0 ± 0.0	93.3 ± 5.8	100.0 ± 0.0	100.0 ± 0.0	40.0 ± 10.0	88.9 ± 2.5

C-3 Randomization Settings

Object and Pose Randomization. To evaluate zero-shot generalization beyond the configurations used in the original tasks, we replace the original object in Tasks 1.1–1.3 with five unseen objects and evaluate each method under three randomized configurations for each object. Object positions and poses are independently varied within the 60×60 cm workspace while preserving the task semantics and success criteria.

Importantly, the demonstrations used to fine-tune the VLA baselines are themselves collected with randomized object positions and poses. Nevertheless, as reported in Table 3, the fine-tuned VLA baselines still exhibit a substantial performance drop when evaluated on unseen objects and configurations, whereas EASG largely preserves its zero-shot performance.

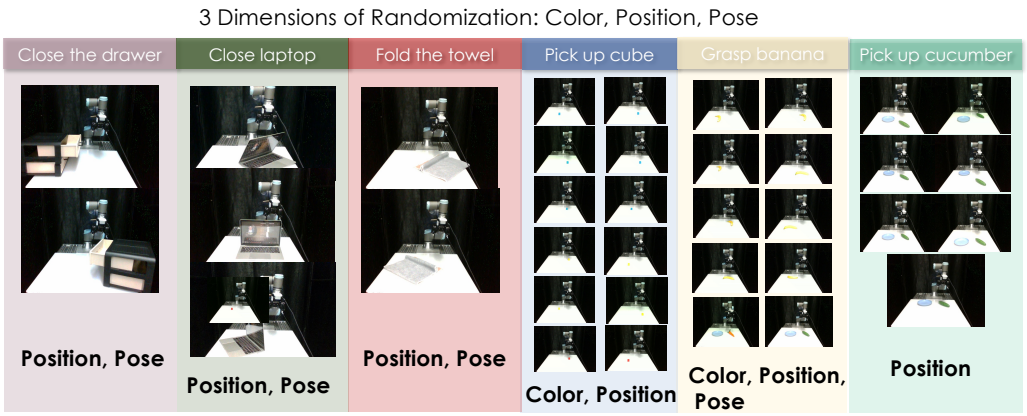

Figure 9: **Randomization settings for real-world generalization experiments.** For each task, we vary both the object instance and its initial position to evaluate the robustness of EASG beyond the original zero-shot setting.

D Fine-tuning Protocols of VLMs with EASG

D-1 Tasks and Metrics of Simulation Benchmarks

We evaluate VLMs post-training with EASG on **LIBERO** (Liu et al., 2023). Their task compositions cover different aspects of visuomotor manipulation and therefore provide a diverse testbed for evaluating continuous-action fine-tuning.

LIBERO. LIBERO organizes manipulation tasks into four complementary suites: **Spatial**, **Object**, **Goal**, and **Long** (LIBERO-10).

Task Success Metrics. For LIBERO we report **episode-level task success rate** as the primary evaluation metric. In LIBERO, an episode is considered successful once the language-conditioned task goal specified by the environment is satisfied, and success rate is computed as the fraction of successful evaluation rollouts.

D-2 Tasks and Metrics of Real-world Scenes

Task Composition Analysis. All fine-tuning tasks are intentionally designed to be more challenging than their zero-shot counterparts, targeting settings that are difficult to complete zero-shot or would otherwise require a very long time to complete in a zero-shot way. The full task compositions and success criteria are summarized in Table 10.

Real-world Success Metrics. The task-specific success criteria for all real-world fine-tuning tasks are summarized in Table 10. For each task and method, we conduct three independent evaluation rounds with 10 trials per round, resulting in 30 trials in total. We report the **mean task success rate** $\pm$ **standard deviation** across the three evaluation rounds. Tasks 1.1–1.3 are tasks that VLM with EASG can complete zero-shot directly and are used to evaluate whether fine-tuning can further improve success rates over already available zero-shot capabilities. Tasks 2.1–2.3 evaluate whether *EASG fine-tuning* can help to solve harder tasks beyond the native zero-shot frontier. The complete fine-tuning results are reported in Table 13.

Type	Task	Task description	Success criterion
Deformable	2.1	Fold the T-shirt in half.	The T-shirt is folded with the two sides partially aligned.
Articulated	2.2	Open the laptop.	The laptop lid is opened beyond 60° and remains stable.
Precise	2.3.1	Pick up the ring and put it on one of the pegs.	The ring is placed onto a peg.
	2.3.2	Hang the cup on the peg.	The cup is stably hung on the peg.
Long-horizon	2.4.1	Open the drawer, then put the Rubik's Cube in the drawer.	The drawer is partially opened and the cube is placed inside.
	2.4.2	Put the grapes in the drawer, then close the drawer.	The grapes are placed inside the drawer, and the drawer is closed within 2 cm of full closure.

Figure 10: **Task Group 2: Capability extension tasks (Hard).** See Appendix C-1 for the full object group we included.

Table 13: **Fine-tuning results on harder real-world manipulation tasks.** Each entry reports the **mean task success rate ± standard deviation** over three independent evaluation rounds with 10 trials per round.

Method	2.1	2.2	2.3.1	2.3.2	2.4.1	2.4.2	Avg.
Ours	80.0 ±10.0	63.3 ±11.5	56.7 ±15.3	43.3 ±15.3	76.7 ±5.8	83.3 ±11.5	**67.2 ±5.1**
π_0-FAST	83.3 ±5.8	70.0 ±10.0	53.3 ±15.3	50.0 ±10.0	80.0 ±10.0	83.3 ±5.8	70.0 ±3.3
OpenVLA	70.0 ±10.0	46.7 ±15.3	43.3 ±11.5	20.0 ±10.0	73.3 ±11.5	76.7 ±5.8	55.0 ±3.3
Octo	73.3 ±5.8	40.0 ±10.0	46.7 ±11.5	26.7 ±15.3	80.0 ±10.0	76.7 ±15.3	57.2 ±4.2

D-3 Real-world Training Protocol

Data Collection Setups. Real-world demonstrations are collected on a **UR3 robot** equipped with a **Robotiq 2F-85 gripper**. We use a **Haptic Touch** device for teleoperation, an **Intel RealSense D435i** as the external scene camera, and an **Orbbec Gemini 305** mounted on the robot wrist for close-range visual observations. During teleoperation, we synchronously record RGB observations, robot proprioceptive states, end-effector poses, and gripper commands at 20 Hz. The operator controls the end-effector through teleoperation mapping. Recorded trajectories are subsequently filtered and converted into the camera-projected, TCP-centered action representation used by EASG for fine-tuning. We collect 30 demonstrations in the real-world fine-tuning experiments **(Task Group 2)** per task unless otherwise specified.

Representative Fine-tuning Inference Trajectories. To qualitatively illustrate the behaviors acquired through post-training, we visualize representative successful inference trajectories of our fine-tuned Qwen3.8-27B by Lora on each **Task Group 2** task in Figure 11. These tasks are designed beyond the model's native zero-shot capability frontier and require more precise spatial reasoning, object orientation control, or longer-horizon action sequence generation. The trajectories show that, after fine-tuning with EASG, the VLM can reliably complete manipulation tasks beyond their zero-shot capabilities.

E Cross-Embodiment

E-1 Experiment Design

Robot Platforms. In addition to the UR3 platform described in Appendix D-3, we evaluate EASG on a second robot embodiment from Flexiv. Both platforms use the same Robotiq 2F-85 gripper, while differing substantially in arm morphology, kinematics, and low-level control. This setting

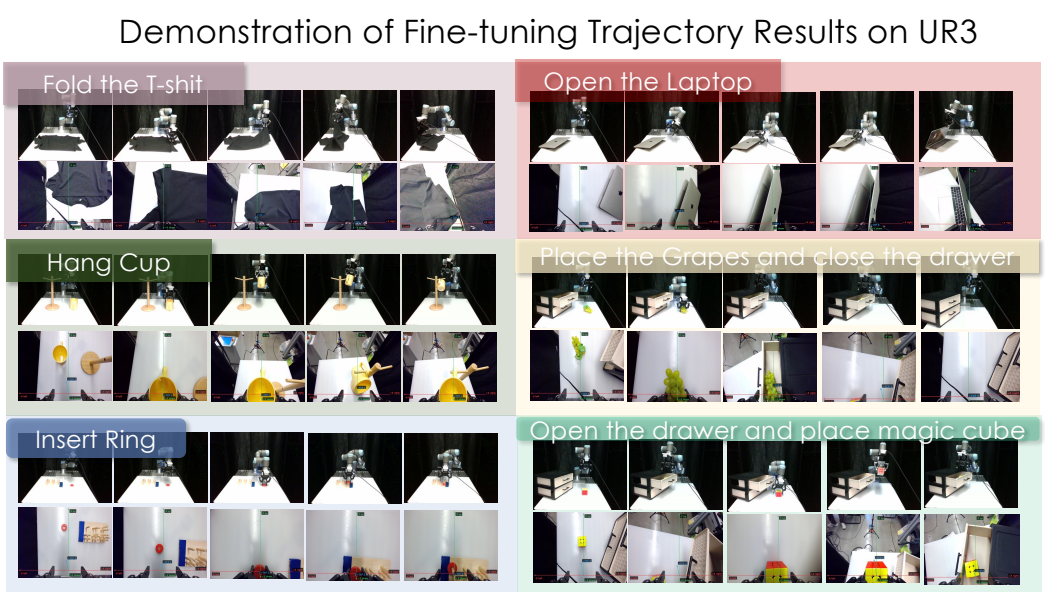

Figure 11: **Representative inference trajectories after EASG fine-tuning.** We show one successful rollout for each Task Group 2 task using the fine-tuned Qwen-3.8-27B. These tasks are beyond the model's native zero-shot capability and are solved after continuous-action fine-tuning with EASG.

allows us to isolate whether the action representation learned by EASG transfers across robot embodiments rather than overfitting to the kinematics of a particular arm.

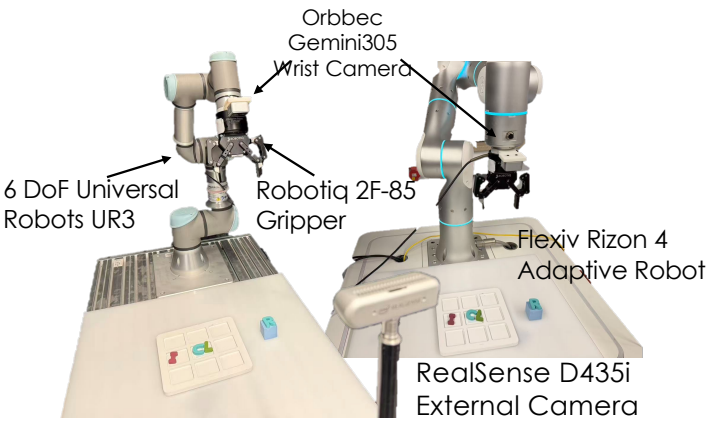

Figure 12: **Robot platforms used for cross-embodiment evaluation.** We compare the UR3 platform used for fine-tuning with a Flexiv robot arm equipped with the same Robotiq 2F-85 gripper. The two robots differ in embodiment and kinematics, while evaluation is initialized from matched end-effector poses. The external camera is only used for recording, **not for inference**.

Experiment Design. A central motivation of EASG is that the VLM predicts **end-effector poses in a task-space representation**, rather than embodiment-specific joint commands. We therefore test whether the same fine-tuned checkpoint can be transferred directly from UR3 to Flexiv without any additional fine-tuning or embodiment-specific robot demonstrations. For each Task in Table 10, the two robots are initialized with matched end-effector poses relative to the workspace, and the same VLM checkpoint and action-grounding interface are used for inference.

Cross-Embodiment Performance. We follow the same real-world evaluation protocol described in Appendix D-3, with three independent rounds of 10 trials per task. Table 14 reports the **mean success rate ± standard deviation** on the original UR3 embodiment and the unseen Flexiv embodiment.

Table 14: **Cross-embodiment transfer of the same fine-tuned EASG checkpoint.** We report mean ± standard deviation over 3 evaluation rounds, with 10 trials per task in each round. No additional fine-tuning is performed for the Flexiv embodiment.

Embodiment	2.1	2.2	2.3.1	2.3.2	2.4.1	2.4.2	Avg.
UR3	76.7±11.5	60.0±10.0	53.3±11.5	36.7±15.3	76.7±5.8	80.0±10.0	**63.9±5.1**
Flexiv	73.3±11.5	56.7±15.3	50.0±10.0	40.0±17.3	73.3±5.8	80.0±17.3	**62.2±3.5**



Figure 13: **Representative synchronized rollouts across robot embodiments.** We visualize the same manipulation task executed by UR3 and Flexiv using the identical fine-tuned EASG checkpoint. Starting from matched end-effector poses, both robots receive pose predictions from the same VLM, while embodiment-specific inverse kinematics and low-level controllers execute the predicted task-space actions.

F Full Analysis of Ablation Study

F-1 What Enables Effective Zero-Shot Action-Space Grounding?

All zero-shot ablations in this section are conducted on **Task Group 1** under the same robot setup, VLM backbone, prompting protocol, and inference configuration. We modify only the corresponding grounding component to isolate its contribution.

Action-Frame Grounding. The proposed EASG represents actions in a **TCP-centered coordinate frame**, such that the predicted motion is expressed relative to the controllable end effector. To examine whether this choice is important, we replace the TCP-centered frame with two alternative coordinate systems: a **robot-base-centered frame** and a **joint-centered frame**, while keeping all other components unchanged. This comparison tests whether the zero-shot capability arises specif-

ically from aligning the action representation with the locally controllable tool frame, rather than merely exposing an arbitrary robot coordinate system to the VLM.

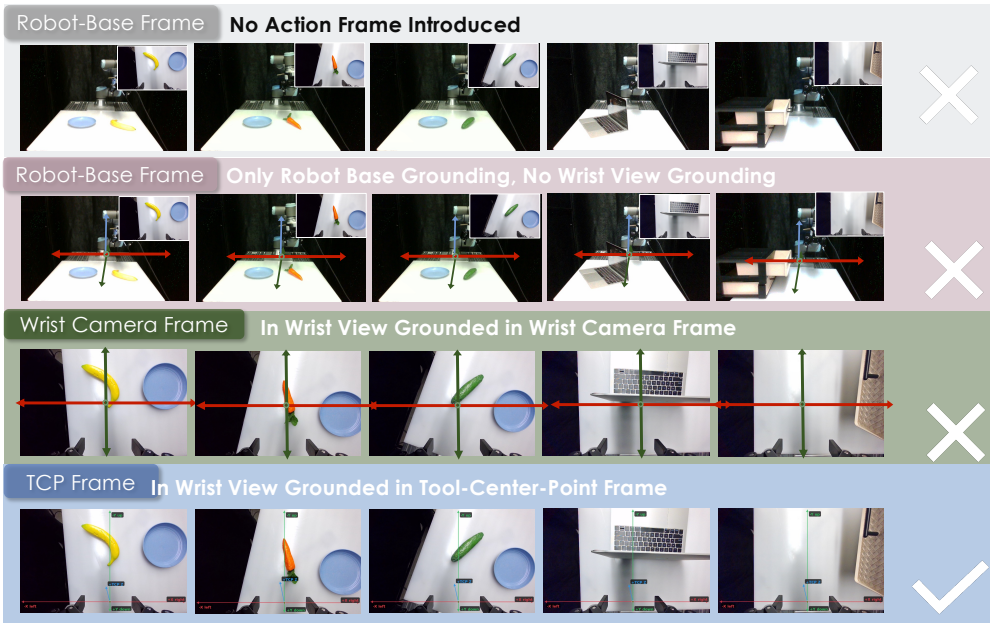

Figure 14: **Qualitative comparison of action-frame grounding.** We visualize representative rollouts of the full TCP-centered grounding and two alternative coordinate-frame choices on the same zero-shot manipulation task.