AxisGuide: Grounding Robot Action Coordinate System in RGB Observations for Robust Visuomotor Manipulation

Jiyun Jang[1], Yujin Sung[1*], Woosung Joung[1*], Daewon Chae[2], Sangwon Lee[3], Sohwi Kim[3], Jinkyu Kim[1,4†], Jungbeom Lee[1†]

[1]Korea University [2]University of Michigan [3]KT R&D Center [4]Kakao Mobility
[*]Equal contribution [†]Co-corresponding authors

Project page: https://jiyunjang-24.github.io/AxisGuide-project/

***Abstract*—Visuomotor manipulation policies trained via large-scale behavior cloning have achieved strong semantic scene understanding, yet often fail to reliably execute correct low-level actions under distribution shifts. For example, even in a simple pick-up task with identical scene layouts, camera viewpoints, and illumination, performance can degrade substantially when the object is placed at unseen locations. We argue that this gap arises from insufficient action understanding, namely the inability to interpret the robot's base-frame action coordinate system in image space. To address this issue, we introduce AxisGuide, a lightweight guidance method that bridges semantic scene understanding and action-coordinate interpretation. Using camera parameters and end-effector poses, AxisGuide renders the robot base-frame axes in each camera view and augments RGB observations with a small set of cue channels that explicitly visualize the meaning of $+x/+y/+z$ motions in image space. Extensive evaluations in both the LIBERO simulation and real-world environments demonstrate that AxisGuide yields substantial performance gains and improved generalization, highlighting the effectiveness of explicit action-coordinate cues for learning reliable and transferable generalist visuomotor policies.**

I. Introduction

Recent visuomotor manipulation policies [3, 16, 22] have achieved impressive performance by scaling behavior cloning with large collections of demonstrations. Building on this foundation, Vision-Language-Action (VLA) models extend visuomotor learning by leveraging vision-language pretraining to improve semantic understanding, enabling broader generalization across tasks [1, 14, 15, 23, 31]. Nevertheless, we still observe a persistent gap between understanding and executing: even in situations where a model appears to semantically "understand" scenes, it often fails to reliably translate that understanding into correct low-level actions [5]. In other words, the policy may capture *what* to do at a high level, but it often struggles with *how* to execute it robustly. This gap naturally leads to a fundamental question: **Do visuomotor policies truly understand actions?**

In manipulation, "understanding actions" is closely tied to understanding the *action coordinate system*. Most modern visuomotor policies perceive the world through RGB images and infer actions from pixels. In most setups [2, 14, 31], actions are defined as relative end-effector (EEF) poses with respect to the robot base frame, typically parameterized as 6D $(x, y, z,$ roll, pitch, yaw) plus a gripper command. Because a visuomotor policy infers actions from pixels and a 6D action is composed of independent unit dimensions, successful task

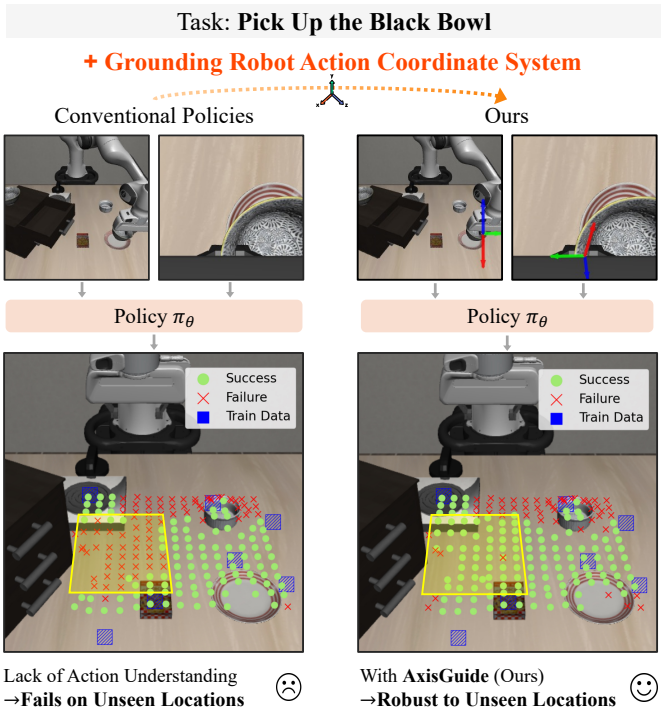


Fig. 1: **AxisGuide: Grounding Robot Action Coordinate System for Robust Manipulation.** Conventional visuomotor policies (left) struggle to generalize beyond training data (blue squares), often failing at unseen locations (yellow box). In contrast, AxisGuide (right) enables robust task execution across a wide range of unseen spatial configurations. By explicitly associating the action space with image observations through grounding the robot action coordinate system in image space, AxisGuide bridges the gap between semantic scene understanding and action coordinate interpretation.

execution requires understanding what each action dimension means in image and how to combine them to generate appropriate behavior in novel situations. Shortly, a policy must recognize how the action coordinate system manifests in the RGB observation to behave robustly. For example, the model must know which direction in the RGB image corresponds to $+x$ in the action space and which rotation corresponds to yaw in the given camera view, and use this knowledge to execute the intended actions.

However, it is inherently difficult for policies to acquire such correspondence between RGB image and action space under the standard observation-to-action setting, because RGB images and robot states rarely provide an explicit direct reference that anchors the action coordinate system. The only available reference for interpreting the action coordinate system in the image is the robot's base, yet it still does not provide an explicit visual cue to the intended semantics. Moreover, the robot's base is often outside the camera's field of view or partially occluded in many robotics datasets [13, 21, 26], making it considerably more difficult to refer to the robot's base frame. Therefore, behavior cloning often degenerates into memorizing correspondences between images and numeric action vectors, which leads to poor generalization.

To illustrate this degeneration, we train a VLA model [23] on a simple pick-up task with randomized object placements, using both a wrist-mounted and a front camera in a setup where the robot base frame is clearly visible, while keeping the instruction and manipulation objective fixed and varying only the object position. We then evaluate whether the policy can pick up the same object at unseen locations (Fig. 1, left). This controlled setting minimizes confounding effects from task semantics and enables a focused evaluation of action understanding, since successful generalization requires the policy to compose new relative end-effector displacements in response to target shifts. Despite consistent object appearance, the model often fails at unseen locations, unable to adjust its end-effector motion to the shifted target. This limitation is also reported in concurrent work [6], which shows that regardless of models, the performance degrades substantially when target object placements are altered. These observations indicate that without understanding the action coordinate system in image space, the policy struggles to generalize beyond the training distribution.

This motivates providing the policy with an explicit association between the action spaces and image observations. Therefore, we propose **AxisGuide**, a lightweight guidance approach that explicitly bridges the action space and its meaning in RGB observations. Using camera parameters, we render the axes of the robot base frame in each camera view and augment each RGB observation with a small set of additional channels that encode these visual cues, as illustrated in Fig. 2. Concretely, for each view we visualize "what $+x/+y/+z$ motion means in the image" by projecting three unit base-frame translations onto the image plane. We take unit translation vectors $(\Delta x, \Delta y, \Delta z) \in \{(1,0,0),(0,1,0),(0,0,1)\}$ from the action space and render their corresponding 2D direction vectors on EEF pose in image space. These vectors are simply computed from camera parameters and the end-effector pose without requiring depth, and then concatenated with the original RGB image in a channel-wise manner as input to the policy. As illustrated in Fig. 1 (right), by making the meaning of the action space explicit in pixels, AxisGuide enables the policy to adapt its end-effector motion more reliably when the object is placed at unseen locations, achieving higher success rate than the observation-only model. Quantitatively, AxisGuide yields substantial performance gains, improving success rates at novel object locations up to 20%p across both real-world and simulation [18] experiments. We further observe consistent success rate improvements in both multi-view and single-view camera configurations, suggesting that explicit action-coordinate grounding leads to more reliable, goal-directed execution.

In summary, our contributions are as follows:

- We identify a previously underexplored gap between semantic understanding and action execution in current visuomotor policies, and define it as the challenge of understanding action coordinate systems in image space.
- We propose AxisGuide, a simple yet effective solution that injects explicit references for the action coordinate systems in RGB images into visual observations.
- We demonstrate consistent gains in performance and generalization across both simulated and real-world manipulation tasks under diverse camera configurations.

II. Related work

A. Visuomotor Policies for Manipulation

Recent work has substantially advanced visuomotor policies for manipulation that map RGB observations directly to actions. Diffusion Policy [3] formulates action generation as a stochastic denoising process over trajectories to capture multimodal action distributions. Vision-Language-Action (VLA) models [2, 14, 31] improve generalization to novel objects and semantically varied instructions by conditioning on language.

One way to represent robot actions is to use the joint space, where actions are defined over joint configurations and a low-level controller supplies the required dynamics. However, such actions are difficult to align with image observations because individual joints are not explicitly visible. An alternative, adopted by most open-source robotic manipulation datasets [13, 21], is to use relative end-effector displacements defined in the Cartesian space of the robot base frame. Following prevalent practice [2, 14, 31], we train policy models to predict relative Cartesian displacements of the end effector.

Under this relative coordinate system, we investigate whether providing a direct visual reference for the action coordinate system in image space alleviates the difficulty of learning to identify the robot's base frame from images.

B. Additional Inputs for Visuomotor Policies

Visuomotor policies are typically trained to predict actions from RGB observations and a task specification, yet this input alone frequently fails to yield robust generalization. Recent

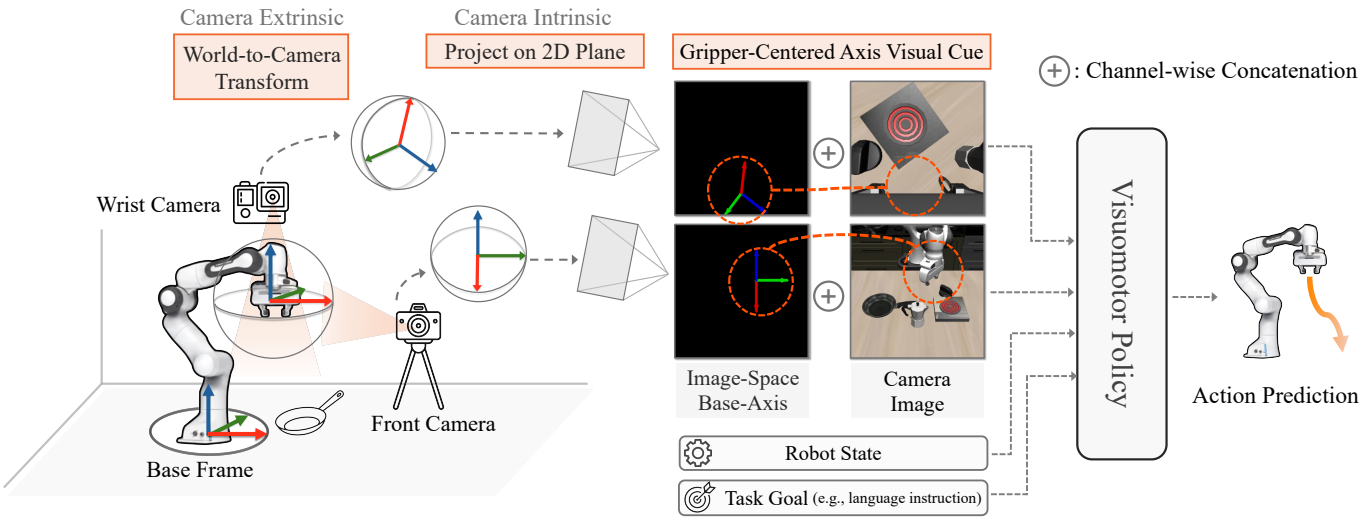


Fig. 2: **An overview of AxisGuide.** Using camera intrinsics and extrinsics, AxisGuide projects the robot base-frame x, y, and z axes onto the 2D image plane, centered at the gripper, and renders them as additional channels alongside RGB images from all cameras. This explicit visualization enables the policy to better understand the correspondence between visual observations and robot base-frame actions.

work therefore introduces additional inputs that act as visual intermediaries or auxiliary context. RT-Trajectory [8] replaces ambiguous language with coarse 2D trajectory sketches to better specify intent and improve generalization. RoboPoint [27] uses a VLM to predict keypoint-like affordances on the image and leverages them to guide low-level control. Several methods also directly augment pixels with extra spatial or temporal cues. To compensate for the lack of temporal and spatial information in image observations, TraceVLA [30] overlays tracked motion traces to encode recent history. AimBot [4] renders reticles and lines using depth and camera geometry providing explicit spatial cues about the object-EEF relationship, and HAMSTER [17] overlays intermediate 2D paths predicted by VLMs to separate high-level planning from motor execution. Complementary to these, other approaches [12, 29] condition policies on hardware context by soft-prompting embodiment-specific configurations or explicitly providing camera geometry to improve robustness across embodiments and sensor setups.

While these techniques enrich observations with task intent, spatiotemporal context, or hardware information, they still do not explicitly consider how the policy's action coordinate system (e.g., "+x translation" or "+z rotation" in the robot frame) should be interpreted in the given image space. AxisGuide addresses this missing component by augmenting RGB observations with an explicit visualization of the action coordinate system in the 2D image space, thereby grounding action semantics in pixel space. Different from prior input-augmentation strategies, Our work explicitly injects action coordinate system information as an image-space cue and represents a new direction for improving the generalization of visuomotor policies.

III. Methodology

A. Preliminaries

Coordinate Transforms. Given the camera extrinsics $\mathbf{T}_{c\leftarrow w} \in \mathbb{R}^{4\times4}$ (world-to-camera), we represent the correspondence between a 3D point in the world frame $\mathbf{p}_w \in \mathbb{R}^3$ and the one in the camera frame $\mathbf{p}_c \in \mathbb{R}^3$ as

$$\begin{bmatrix}\mathbf{p}_c\\1\end{bmatrix} = \mathbf{T}_{c\leftarrow w}\begin{bmatrix}\mathbf{p}_w\\1\end{bmatrix}. \tag{1}$$

We assume a pinhole camera model [7] with intrinsics $\mathbf{K}$, under which a 3D point $\mathbf{p}_c = (X, Y, Z)$ with $Z > 0$ is projected onto the image plane as $\mathbf{p}_i = (u, v)$, where

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{aligned} u &= f_x \frac{X}{Z} + c_x, \\ v &= f_y \frac{Y}{Z} + c_y. \end{aligned} \tag{2}$$

Policy Learning. Visuomotor policies are commonly trained via behavior cloning [24] on robot demonstration datasets $\mathcal{D}$ by maximizing the likelihood of expert actions conditioned on the current observation. Concretely, letting the policy π_θ predict either a single action a_t or a horizon-H action chunk $a_{t:t+H}$, we optimize

$$\max_\theta \mathbb{E}_{(a_{t:t+H},\, o_t,\, c)\sim\mathcal{D}}\Big[\log \pi_\theta(a_{t:t+H} \mid o_t, c)\Big]. \tag{3}$$

$o_t = (I_t, q_t)$ denotes the observation at time t, where I_t is the image input (single-view or multi-view) and q_t is the proprioceptive state (e.g., joint positions). The conditioning variable c is optional and depends on the policy. It may include a natural-language instruction ℓ for vision-language-action (VLA) models [1, 14, 15], or be omitted for vision-based visuomotor policies [3].

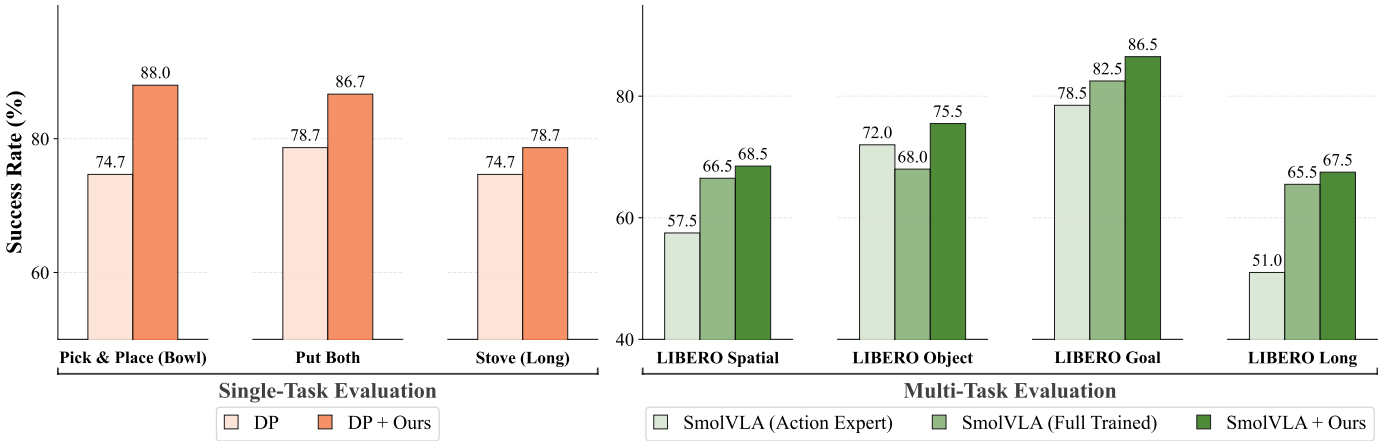


Fig. 3: **Quantitative Results in the Multi-View Simulation Setup (LIBERO).** We compare success rates of AxisGuide with baseline methods in single-task (left) and multi-task (right) settings using wrist and front cameras. Unlike the standard SmolVLA training pipeline [23], we train the full model including the image backbone to support additional coordinate cue channels. For fair comparison, we report both the action-expert-only and fully trained variants of SmolVLA. AxisGuide consistently outperforms Diffusion Policy (DP) [3] and SmolVLA across all evaluated tasks.

B. Method Overview

A key challenge in learning image-to-action policies is that the action space is defined in a robot-centric coordinate system (e.g., the robot base frame), while the observation is an RGB image and proprioceptive state that provide no explicit reference for how each action dimension (x, y, z) should be interpreted in the current view. As a result, the policy must implicitly infer the mapping between 6D robot actions and their visual consequences in image space purely from data, which can lead to poor generalizability under distribution shifts. AxisGuide addresses this mismatch by making the action coordinate system visually explicit, as illustrated in Fig. 2. We project the robot base-frame axes into the image space and inject them as an additional visual cue, so that the policy can directly read how "move along $+x$" or "move along $+z$" appears in the current camera view. Concretely, AxisGuide augments each camera image observation with a minimal, explicit visualizations of the action coordinate system. At every timestep, we render three *direction vectors* originating from the current end-effector location in the image. These vectors correspond to unit translations along the robot base frame's x, y, and z axes. We encode these vectors as a $3 \times H \times W$ action coordinate cue image A_t, where x-, y-, z-axis directions are represented in the red, green, blue channels of an RGB image, respectively. This cue image is concatenated channel-wise to the original RGB input and provided to the policy. In the next section, we provide a detailed description of the rendering process for A_t.

C. Computing Action Coordinate Cue Image.

Let $\mathbf{p}_w \in \mathbb{R}^3$ be the current gripper position in the robot-base/world frame, and let $\Pi(\cdot)$ be the camera projection defined by the intrinsics/extrinsics following Eq. (1) and Eq. (2). We first obtain the pixel location of the gripper by projecting $\mathbf{p}_w$ onto the image plane, $\mathbf{o} = \Pi(\mathbf{p}_w) \in \mathbb{R}^2$. To compute image-space direction vectors corresponding to unit translations along each base axis, we consider three canonical unit moves $\Delta\mathbf{p}_w^{(x)} = \varepsilon\mathbf{e}_x$, $\Delta\mathbf{p}_w^{(y)} = \varepsilon\mathbf{e}_y$, $\Delta\mathbf{p}_w^{(z)} = \varepsilon\mathbf{e}_z$, where $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the robot base-frame basis vectors and $\varepsilon > 0$ is a small step size. We project the translated points onto the image plane and compute the corresponding image space displacements as

$$\Delta\mathbf{u}^{(k)} = \Pi\left(\mathbf{p}_w + \Delta\mathbf{p}_w^{(k)}\right) - \Pi(\mathbf{p}_w), k \in \{x, y, z\}. \quad (4)$$

Then we normalize them to obtain unit image-space direction vectors $\hat{\mathbf{d}}^{(k)}$. Finally, we render three arrows starting at $\mathbf{o}$ and pointing along $\hat{\mathbf{d}}^{(x)}, \hat{\mathbf{d}}^{(y)}, \hat{\mathbf{d}}^{(z)}$, using red/green/blue channels of an RGB image for $x/y/z$, respectively. We anchor these arrows at $\mathbf{o}$ (the projected end-effector pixel) to emphasize the gripper's current interaction point and to provide an end-effector-centric directional reference. Rendering the $x/y/z$ directions at the moving EEF explicitly visualizes where the robot will move from its current state in the image, making the action semantics directly observable. The rendered result forms an action coordinate cue image $A_t \in \mathbb{R}^{3\times H\times W}$.

D. Policy Learning With Action Coordinate Cue Image.

We inject the action-coordinate cue image via channel-wise concatenation rather than overlaying them onto the RGB image observation. Overlays can occlude small but critical visual details and alter the raw appearance distribution, which may interfere with perception. Channel-wise concatenation preserves the original RGB content while providing a dedicated cue channel for the policy to exploit. Following the common practice of injecting additional visual tokens/channels by early fusion [8, 12], AxisGuide also augments each view by concatenating the RGB image with the action-coordinate cue image channel-wisely:

$$\tilde{I}_t^{(v)} = [I_t^{(v)}; A_t^{(v)}] \in \mathbb{R}^{6\times H\times W}, \; \tilde{o}_t = \left(\{\tilde{I}_t^{(v)}\}_{v=1}^{V}, q_t\right), \quad (5)$$

where v indexes camera views. To process $\tilde{I}_t^{(v)}$, we simply increase the input channel dimension of the vision backbone [9, 25, 28]'s first convolution layer from 3 to 6 (per view), while keeping the rest of the backbone and the policy head unchanged. We then train the policy with the standard behavior cloning objective from Eq. (3), replacing o_t with $\tilde{o}_t$:

$$\max_{\theta} \mathbb{E}_{(a_{t:t+H},\, \tilde{o}_t,\, c)\sim\mathcal{D}} \Big[\log \pi_\theta(a_{t:t+H} \mid \tilde{o}_t, c) \Big]. \quad (6)$$

IV. Experiment

In this section, we present multiple experiments to evaluate the effectiveness of AxisGuide with detailed analysis addressing the following key research questions:

- Q1. (Section IV-B) Does AxisGuide help the policy understand action coordinate systems?
- Q2. (Section IV-C) Does AxisGuide improve control performance and task success rates?
- Q3. (Section IV-D) Does AxisGuide help the policy better exploit multi-view observations for composing intended actions?
- Q4. (Section IV-E) How does AxisGuide compare with existing visual cue baselines?
- Q5. (Section IV-F) Is AxisGuide robust to calibration errors in real-world systems?
- Q6. (Section IV-G) What design choices are important for effective AxisGuide representations?

A. Experimental Setup

Simulation Setup. We conduct all simulation experiments on the LIBERO benchmark [18], which provides four task suites for studying lifelong learning in robotic manipulation: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each suite stresses a different axis of generalization. LIBERO-Spatial varies the scene layout while keeping the object set fixed, emphasizing spatial reasoning under changing geometries. LIBERO-Object varies the object instances within a fixed layout, testing whether a policy can recognize and manipulate diverse objects under consistent scene structure. LIBERO-Goal keeps objects and layouts fixed but changes task goals, evaluating goal-directed behavior under shared physical context. LIBERO-Long contains long-horizon manipulation tasks with diverse objects, layouts, and goals, and is the most challenging suite as success depends on accurate gripper-object alignment over extended trajectories. For training data generation, we follow prior work [14] and regenerate all demonstrations while rendering RGB observations with action-coordinate cue images. All the images are resized to 256×256 pixels. Additional details on task definitions and data collection are provided in the supplementary material.

Real-world Setup. We conduct real-world experiments with a UR5e robot and an RG2 gripper in a tabletop environment with one front RGB camera. For the multi-view setup, we attach a wrist-mounted camera using the UR5e wrist mount design from [20]. For all the real-world experiments in Sec IV-C and Sec IV-D, we select (1) **Pick & Place (Grape)**: picking up the grape and placing it on the plate, (2) **Flip Pot**: flipping the pot upright, and (3) **Close Pot**: picking up the pot lid and closing the pot as shown in Fig. 4. **Pick & Place (Grape)** measures basic grasping and placement accuracy by positioning objects in various locations across episodes. **Flip Pot** stresses object pose understanding and rotational control, and **Close Pot** assesses fine-grained gripper-object alignment in a contact-rich setting. Further details are provided in supplementary material.

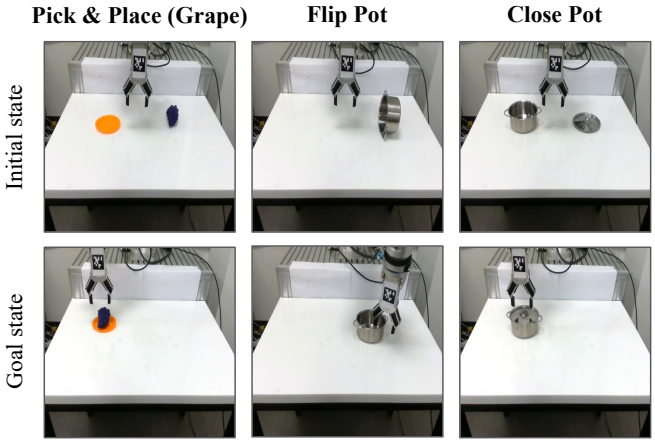
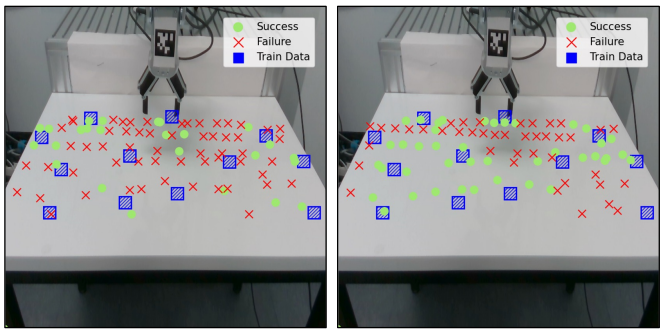


Fig. 4: **Real-world Manipulation Tasks for Evaluation.** We show initial states (top) and goal states (bottom) for **Pick & Place (Grape)**, **Flip Pot**, and **Close Pot**, which require different combinations of translational and rotational actions.

Policies and Baselines. For single-task experiments, we adopt Diffusion Policy (**DP**) [3] as our base visuomotor policy. For multi-task experiments, we use **SmolVLA** [23] to train a Vision-Language-Action model. Notably, SmolVLA is typically trained by freezing the pretrained vision-language backbone and optimizing only the action expert. In contrast, AxisGuide requires adapting the visual representation to effectively apply the additional coordinate cue channels. For fair comparison, we train the full model including the image backbone. This is a standard and often necessary choice when training VLAs, where adapting the visual encoder is crucial for grounding action prediction in task- and domain-specific visual features, as done in [1, 11, 14]. All models are trained with identical inputs and pipelines, differing only in the use of coordinate cue images.

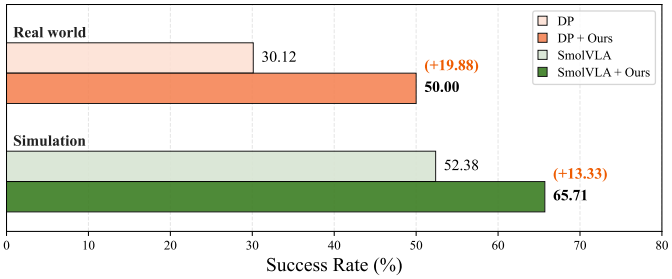
B. Does AxisGuide Help the Policy Understand Action Coordinate Systems?

In this section, we evaluate whether AxisGuide encourages policies to learn action semantics from visual observations. To test whether a policy correctly interprets the meaning of actions, we construct a dataset in which the same object is placed at multiple locations. To only evaluate action-coordinate understanding for evaluation while minimizing confounding factors such as long-horizon planning and complex object interactions, we adopt **Pick Up** as a simple manipulation task. If a policy truly learns the task objective and the semantics of its actions, it should successfully pick up the object regardless of its location. We collect demonstrations across diverse object



DP DP + AxisGuide (Ours)

(a) Spatial Generalization Across Object Positions in Real World



(b) Comparison of Success Rates in Real-World and Simulation

Fig. 5: **Generalization to Novel Object Positions.** (a) shows that the baseline DP (left) generalizes poorly, with success largely confined to regions near training data, whereas DP with AxisGuide (right) reliably reaches unseen object positions between clusters, which is consistent with the simulation results in Fig. 1. We report corresponding success rates in (b), where AxisGuide delivers substantial gains in both real-world and simulation settings.

positions under a multi-view configuration using both front and wrist-mounted cameras.

In simulation, we use the official LIBERO-Spatial dataset, which involves picking up a black bowl from varying locations, and evaluate performance on the **Pick Up (Bowl)** task. We collect 400 demonstrations from the LIBERO-Spatial suite and report performance as the average success rate over 210 evaluation rollouts on unseen locations, as illustrated in Fig. 1. In the real world, using the corresponding setup shown in Fig. 5 (a), we evaluate DP on the **Pick Up (Pear)** task to verify that the observed effects are not model-specific. We collect 120 demonstrations and evaluate performance over 84 rollouts on unseen locations. Further details are provided in the supplementary material.

The quantitative results in Fig. 5 (b) suggest that AxisGuide enables policies to better reach and contact objects not only at seen positions but also at novel object locations. Specifically, AxisGuide improves the simulation success rate from 52.38% to 65.71% (+13.33%p) and the real-world success rate from 30.12% to 50.00% (+19.88%p). Qualitative results in Fig. 1 and Fig. 5 (a) further show that baseline models generalize poorly beyond the training data, whereas models trained with AxisGuide reliably reach unseen object positions between clusters in both simulation and real-world settings.

To better understand these results, we further visualize representative rollout trajectories from the evaluation set (Fig. 6). In real-world and simulation, SmolVLA exhibits weaker robustness to object location shifts, often drifting toward a training-like configuration even when the bowl is relocated, whereas AxisGuide produces a trajectory that redirects toward the shifted target and reaches it more precisely. Overall, these results indicate that AxisGuide encourages policies to produce appropriate action adjustments by grounding action coordinates in pixels, leading to more robust behavior under distribution shift.

TABLE I: **Quantitative Comparison of Single-View Visuomotor Policies in the LIBERO Simulation.** We evaluate average success rates (%) over 75 rollouts (25 rollouts × 3 seeds). See the Sec IV-C for further discussion.

Method	**Pick & Place (Bowl)**	**Drawer**	**Stove**
DP [3]	69.33	86.67	92.00
DP + KYC [12]	73.33	89.33	96.00
DP + Ours	**82.67** (13.34↑)	**93.33** (6.66↑)	**100.0** (8.00↑)

TABLE II: **Quantitative Comparison of Visuomotor Policies in Real-World.** We evaluate average success rates (%) over 30 rollouts. The single-view setup uses a fixed front camera, while the multi-view setup includes an additional wrist-mounted camera. More details about tasks are provided in Sec IV-A.

View type	Method	**Pick & Place (Grape)**	**Flip Pot**	**Close Pot**
Multi-view	DP	93.39	73.33	36.66
	DP + Ours	**96.66** (3.27↑)	**93.33** (20.00↑)	**53.33** (16.67↑)
Single-view	DP	83.33	36.65	33.33
	DP + KYC	86.67	23.33	40.00
	DP + Ours	**93.33** (10.00↑)	**50.00** (13.35↑)	**40.00** (6.67↑)

C. Does AxisGuide Lead to Better Control and Higher Task Success Rate?

In this section, we evaluate whether additional information from AxisGuide's action coordinate cues actually help the policy to successfully execute desired tasks. Since many manipulation systems are limited to a single external camera, we test whether AxisGuide is effective in the front-camera-only setting. As a point of comparison for anchoring action semantics in image space, we adopt **Know Your Camera (KYC)** [12] as an additional baseline in the single-view setup, which explicitly conditions the policy on camera information.

We evaluate AxisGuide in both simulation and the real world using a single-view Diffusion Policy (DP). In simulation, we consider three LIBERO tasks spanning diverse action types and control skills: (1) **Pick & Place (Bowl)**: picking up the black bowl between the plate and the ramekin and placing it on the plate, (2) **Drawer**: opening the middle drawer of the cabinet, and (3) **Stove**: turning on the stove. In the real world, we follow the task setup in Sec. IV-A. For each task, we train

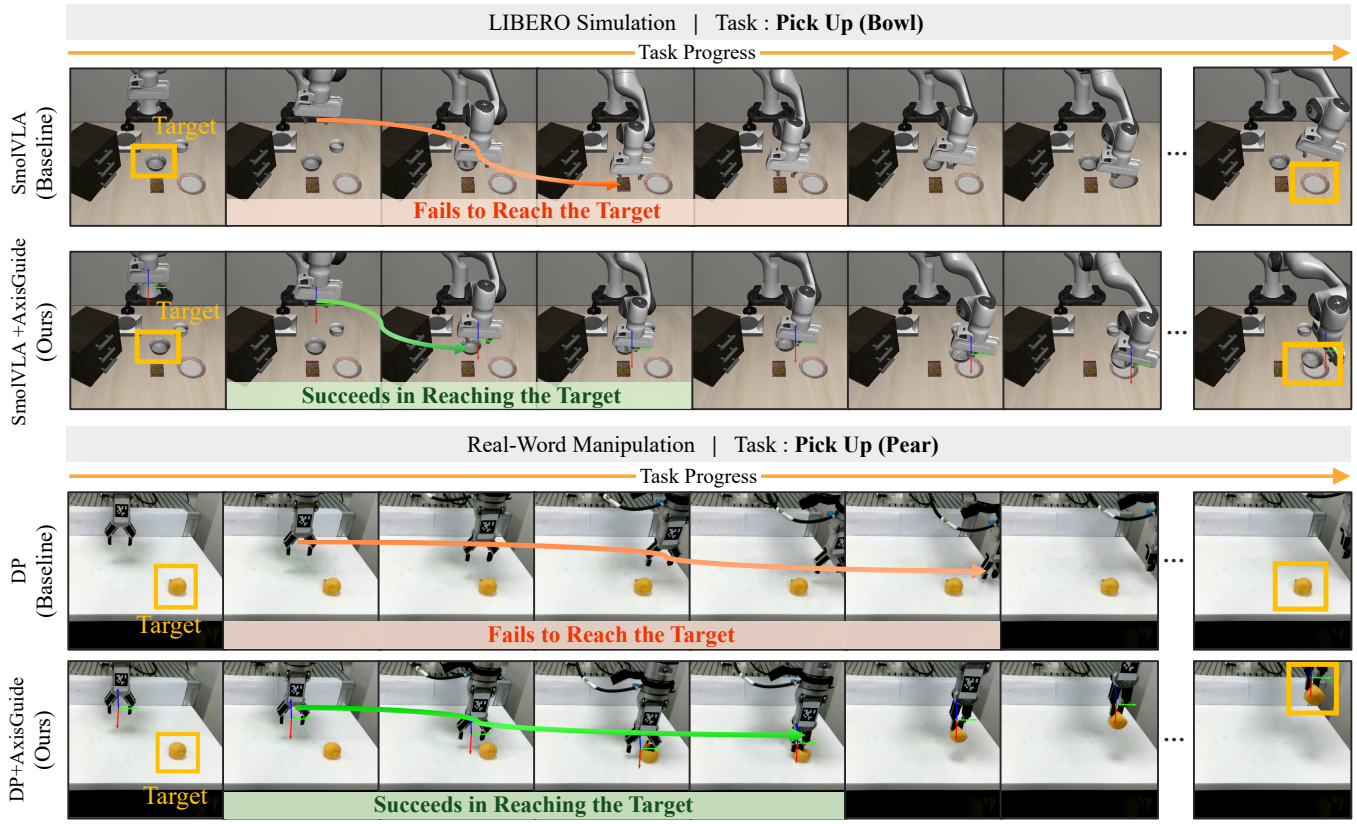


Fig. 6: **Rollout Behaviors Under Unseen Object Locations on LIBERO Simulation and Real-World Manipulation.** In the **Pick Up (Bowl)** task in the LIBERO simulation (top), the baseline model (SmolVLA) fails to adapt its actions when the target bowl is placed at an unseen location. In contrast, the same model trained with AxisGuide precisely reaches the target by grounding the action coordinate system in the image-space. A similar trend is observed in the real-world **Pick Up (Pear)** task (bottom), where DP serves as the baseline.

DP and report the average success rate over 75 evaluation rollouts in simulation (25 rollouts $\times$ 3 seeds) and 30 rollouts in the real world.

As shown in Table I, AxisGuide yields consistent improvements across all simulation tasks, with the largest gain on **Pick & Place (Bowl)**, where success depends on accurately perceiving small position shifts and executing precise corrections. Compared to the RGB-only baseline, AxisGuide improves success rate by 13.34%p and also outperforms KYC [12] by 9.34%p. Also, Table II (Single-view) shows the same trend in the real world, with particularly large gains on **Pick & Place (Grape)** and **Flip Pot**.

Overall, these results suggest that AxisGuide's explicit coordinate reference improves reliable task execution across diverse manipulation behaviors, boosting success in both simulation and real-world settings.

D. Does AxisGuide Help the Policy Better Exploit Multi-View Observations?

We evaluate **AxisGuide** in a multi-view setting with a fixed front camera and a wrist-mounted camera. While the front view provides a relatively consistent correspondence between image-space directions and the robot base-frame axes, the wrist view continuously changes this correspondence as the end-effector moves. As a result, a policy must interpret wrist-view observations in a coordinate-consistent way to act correctly. In this section, we show that AxisGuide remains reliable in this dynamically changing setting by providing an explicit axis reference in each view, enabling robust use of wrist-view observations. We first study a single-task setting to show the benefits of coordinate cues in multi-view control, and then extend to a multi-task setting to test whether the gains persist as task diversity increases. All training and evaluation follow the same setup as Sec. IV-C.

Single-Task Setting. In simulation, we select (1) **Pick & Place (Bowl)**, (2) **Put Both**: putting both the cream cheese box and the butter in the basket (LIBERO-Long), and (3) **Stove (Long)**: turning on the stove and putting the moka pot on it (LIBERO-Long). We choose harder tasks than those in the simulation setting of Sec. IV-C, since adding a wrist-mounted camera already improves success rates without modifying the training pipeline. For real-world evaluation, we follow the real-world task setup in Sec. IV-C.

As shown in Fig. 3, AxisGuide improves success rate on **Pick & Place (Bowl)** from 74.7% to 88.0% (+13.3%p) and on **Put Both** from 78.8% to 86.7% (+7.9%p) in simulation.

In real-world tasks, As summarized in Table II (Multi-view), AxisGuide improves performance on all evaluated real-world tasks, boosting success on **Pick & Place (Grape)** by 3.27%p, **Flip Pot** by 20.00%p, and **Close Pot** by 16.67%p. Overall, we find that AxisGuide yields larger gains on tasks that benefit most from wrist-view, such as **Flip Pot** and **Close Pot**.

Multi-Task Setting. As task diversity scales, a single language-conditioned policy must execute many instruction-following behaviors across diverse object configurations, making robust action-coordinate grounding increasingly important. We therefore evaluate whether AxisGuide remains effective in this multi-task setting. We train a SmolVLA [23] model on 10 tasks from each LIBERO suite (LIBERO-Spatial, LIBERO-Goal, LIBERO-Object, and LIBERO-Long). For a fair comparison, we include two SmolVLA baselines trained with the same pipeline: (i) action-expert-only with a frozen VLM backbone, and (ii) full fine-tuning. We run 20 rollouts per task and report the suite-level average success rate aggregated over all 200 rollouts (10 tasks $\times$ 20 rollouts) in each suite. We observe consistent gains in the multi-task setting across all four LIBERO suites, as shown on the right of Fig 3. Notably, SmolVLA trained with AxisGuide outperforms the fully trained SmolVLA by 7.5% on LIBERO-Object. These results suggest that AxisGuide provides useful action coordinate grounding even when the policy must map diverse language instructions to low-level actions across many tasks. In other words, the gains in the multi-task setting show that AxisGuide scales beyond single-task behavior cloning: it helps the policy maintain a consistent interpretation of action dimensions under increased task diversity, leading to more robust instruction-conditioned execution even in multi-view settings.

E. How Does AxisGuide Compare with Existing Visual Cue Baselines?

We compare our method with prior image-cue baselines [4, 30] on the Novel Object Position task in Fig. 5. TraceVLA [30] and AimBot [4] aim to improve scene understanding by providing visual cues that encode the spatial relationship between the object and the end-effector. However, successful task execution in novel environments requires more than understanding the spatial relationship. The policy must also determine what action to take in the novel scene, which requires understanding the robot's action coordinate system in image space. While TraceVLA and AimBot enrich the observation with scene-level spatial cues, they are not specifically designed to explicitly represent the robot's action coordinate system. As a result, when the object appears in unseen locations, these methods offer limited guidance on how the policy should adjust its actions, making improvements under distribution shift difficult. In contrast, by explicitly grounding the robot's action coordinates in the visual observation and showing where the x, y, and z action directions lie in the image, AxisGuide helps the policy infer how to move toward a target even under novel object positions. This leads to larger gains in Table III and enables more reliable, robust manipulation.

TABLE III: **Comparison with visual cue baselines on the Novel Object Position task.** We report success rate (%) for SmolVLA and its variants augmented with different visual cues.

Method	Success Rate (%)
SmolVLA [23]	52.38
+ TraceVLA [30]	51.90
+ AimBot [4]	52.38
+ AxisGuide (Ours)	**65.71**

F. Is AxisGuide Robust to Calibration Errors in Real-World Systems?

We further evaluate the robustness of our method to projection errors caused by inaccurate system calibrations on the Novel Object Position task in Fig. 5. In the Table IV, we report results under extrinsic (left) and intrinsic (right) perturbations at inference time, while using a model trained with unperturbed camera parameters. Errors in robot kinematics and EEF pose estimations are reflected in this experiment, since they also perturb the projected EEF position. These results demonstrate our method's robustness to realistic calibration noise, as performance degrades marginally and still exceeds the baseline [23] performance of 52.4 reported under unperturbed settings.

TABLE IV: **Calibration sensitivity analysis.** We evaluate robustness to calibration errors by injecting extrinsic perturbations (camera translation and rotation, left) and intrinsic perturbations (focal length and principal point offsets, right) at inference time, while training with unperturbed parameters.

Rot. \ Trans.	0 cm	1 cm	2 cm	3 cm	Focal. \ P.P.	0 px	5 px	10 px
0°	**65.7**	64.3	61.9	62.4	0%	**65.7**	63.8	65.2
3°	64.8	64.8	62.9	61.4	5%	64.3	63.8	65.2
6°	64.3	64.8	64.3	63.3	10%	64.3	64.3	63.8

G. What Design Choices Are Important for Effective AxisGuide Representations?

We ablate three key design choices in AxisGuide: *(i) how* the coordinate cues are injected into the visual input, *(ii) where* the cues are positioned in the image, and *(iii) whether* the projected action directions are normalized. Specifically, we compare (a) overlaying the axes on top of the RGB image versus concatenating the cue as additional input channels, (b) placing the cue in a fixed, center-aligned location versus EEF-aligned positioning, where the cue is anchored to the end-effector location in the image to reflect the action-centric viewpoint, and (c) using raw projected vectors versus normalized unit directions. We evaluate all variants on the **Pick & Place (Bowl)** task using the same Diffusion Policy backbone and training protocol in simulation, and report success rates.

As shown in Table V, adding AxisGuide cues improves performance across all variants, confirming that explicit coordinate grounding is beneficial. However, the magnitude of

TABLE V: **Ablation of AxisGuide design choices.** Average success rates (%) over 75 rollouts on **Pick & Place (Bowl)**. **Inject.** denotes how the visual cue is incorporated into the input (overlay on RGB vs. channel-wise concatenation). **Pos.** indicates how the cue is positioned in the image (eef: aligned with the end-effector; center: image-centered). **Norm.** specifies whether the projected action directions are normalized to unit vectors.

Variant	Inject.	Pos.	Norm.	Success (%)
DP (Baseline)	–	–	–	74.67
DP + AxisGuide (v1)	overlay	eef	yes	84.00 (+9.33)
DP + AxisGuide (v2)	concat	center	yes	80.00 (+5.33)
DP + AxisGuide (v3)	concat	eef	no	85.30 (+10.63)
DP + AxisGuide (Ours)	**concat**	**eef**	**yes**	**88.00 (+13.33)**

improvement depends strongly on the injection and positioning choices. Overlay + EEF-aligned (v1) improves success from 74.67% to 84.00% (+9.33%p), suggesting that even a pure visual overlay can provide useful directional information when it is localized around the region where actions are executed. In contrast, Concat + Center-aligned (v2) achieves 80.00% (+5.33%p), indicating that simply providing coordinate cues is not sufficient. Their spatial placement matters, and placing the cue at the image center can be less informative when the relevant interaction occurs away from the center. Finally, we observe that normalizing the projected action directions further improves performance. Without normalization (v3), the success rate drops to 85.30%, compared to 88.00% with normalization. This suggests that removing scale variations caused by depth and camera geometry allows the policy to focus on directional information, leading to more stable and consistent behavior.

Our final design, Concat + EEF-aligned with normalization, yields the best performance, reaching 88.00% success rate (+13.33%p over the baseline). This shows that how we inject the cue matters: compared to a pure overlay, channel concatenation provides the model with additional cue channels without altering or occluding the original RGB content, enabling the backbone to access both the raw appearance information and the coordinate cues in a clean, disentangled representation. Moreover, anchoring the cue at the end-effector further improves performance by placing the coordinate reference exactly where actions are executed, making the cue most relevant to the local interaction. We additionally find that normalizing the projected action directions further improves performance by removing scale variations, allowing the policy to focus on directional action semantics. Overall, these results motivate our final design choice of injecting AxisGuide cues via concatenated channels with EEF-aligned positioning and normalized directions.

V. Limitations

First, AxisGuide assumes access to camera parameters which are used to project the base-frame axes into each view. While such information is available in many simulation benchmarks and can be measured or calibrated in real-world systems, it may not be readily provided in all datasets or deployment environments. Fortunately, large-scale datasets such as DROID [13] include calibrated multi-view setups, suggesting that this requirement is not prohibitive. Moreover, when calibration metadata is missing, recent camera-to-robot pose estimation methods [10, 19] could be used to estimate the required extrinsics, potentially addressing this limitation.

Second, we observe that even when AxisGuide successfully tracks objects at novel locations, failures can still occur in contact-rich stages such as grasp execution, suggesting that coordinate grounding does not resolve all sources of error. Addressing such failures may require complementary improvements in perception of contacts, gripper-object interactions, or reward signals.

Finally, our evaluation, while spanning multiple tasks and including real-world experiments, is still limited in scale compared to massive robot datasets. A more scalable study on large, diverse datasets (e.g., DROID [13]) would further clarify how AxisGuide behaves under broader distribution shifts and whether the gains persist when training is scaled up.

VI. Conclusion

In this work, we study why visuomotor manipulation policies can exhibit strong semantic scene understanding yet still fail to execute correct low-level actions under distribution shifts. We show that even with identical layouts, viewpoints, and illumination, performance can drop sharply when the object is placed at unseen locations. We argue that a key cause is insufficient action understanding. To address this, we propose **AxisGuide**, a simple but effective method that explicitly visualizes the robot's base-frame action coordinate system in each camera view. By rendering unit base-frame motion directions and providing them as additional input channels, AxisGuide connects visual observations to 3D action semantics without requiring depth, extra supervision, or architectural changes. Across extensive experiments in real-world and simulation, AxisGuide consistently improves task success and robustness under distribution shift (e.g., unseen object locations).

Acknowledgments

We would like to sincerely thank Prof. Sungjoon Choi for providing the robotic platform and supporting the real-world experiments. Daewon Chae is supported by the Hyundai Motor Chung Mong-Koo Foundation. This work was the result of project supported by KT(Korea Telecom)-Korea University AICT R&D Center. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) under the the artificial intelligence star fellowship support program to nurture the best talents (IITP-2026-RS-2025-02304828, 20%) grant, IITP-ICT Creative Consilience Program grant (IITP-2026-RS-2020-II201819, 10%), and by the National Research Foundation of Korea(NRF)(RS-2026-25488668, 10%), funded by the Korea government(MSIT).

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[4] Yinpei Dai, Jayjun Lee, Yichi Zhang, Ziqiao Ma, Jianing Yang, Amir Zadeh, Chuan Li, Nima Fazeli, and Joyce Chai. Aimbot: A simple auxiliary visual cue to enhance spatial awareness of visuomotor policies. In *Conference on Robot Learning*, pages 2409–2429. PMLR, 2025.

[5] Irving Fang, Juexiao Zhang, Shengbang Tong, and Chen Feng. From intention to execution: Probing the generalization boundaries of vision-language-action models. *arXiv preprint arXiv:2506.09930*, 2025. URL https://arxiv.org/abs/2506.09930.

[6] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Liberoplus: In-depth robustness analysis of vision-language-action models, 2025. URL https://arxiv.org/abs/2510.13626.

[7] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. prentice hall professional technical reference, 2002.

[8] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. Vipe: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025.

[11] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[12] 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. *arXiv preprint arXiv:2510.02268*, 2025.

[13] 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, 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. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.120.

[14] 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 Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[15] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[16] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[17] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Representation Learning*, volume 2025, pages 24040–24068, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3bfee3bc6639c36e6e7b058db909f760-Paper-Conference.pdf.

[18] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[19] Jingpei Lu, Zekai Liang, Tristin Xie, Florian Richter, Shan Lin, Sainan Liu, and Michael C Yip. Ctrnet-x: Camera-to-robot pose estimation in real-world conditions using a single camera. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1914–1920. IEEE, 2025.

[20] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 44(4):592–606, 2025.

[21] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[22] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[23] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[24] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4950–4957. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/687.

[25] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[26] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[27] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[28] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[29] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[30] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *The Thirteenth International Conference on Learning Representations*.

[31] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

VII. Supplementary Material

A. Real World Setup Details

Fig. 7 shows the UR5e robot setup used in our real-world experiments. Each real-world task is evaluated to test the policy's ability to handle fine-grained control. All tasks use a 10Hz control frequency with RGB observations resized to 256×256. To verify that AxisGuide transfers to real-world deployment, we design a set of tabletop manipulation tasks that require precise action coordinate grounding and contact-rich control. Below, we describe the task setups in detail, including data collection protocols, randomization ranges, episode lengths, and success criteria.

1) **Pick & Place (Grape)**
   - **Task:** Picking up a grape from a random initial position on the table and placing it onto a target plate.
   - **Dataset:** 50 teleoperated demonstrations. To introduce spatial variety, the *plate* is sampled from a predefined planar region with jittering in the left–right and front–back directions. The *grape* is placed within a specific workspace, varying primarily along the front–back axis relative to the robot.
   - **Evaluation:** 30 trials are conducted. The initial positions of both the grape and the plate are randomized within the same spatial boundaries and perturbation ranges used during the data collection phase.
   - **Evaluation:** 30 trials are conducted. The initial positions of both the grape and the plate are randomized within the same spatial boundaries and perturbation ranges used during the data collection phase.
   - **Episode Length:** 140 timesteps (14 seconds).
   - **Success Criterion:** The task is successful if the grape remains stably on the plate in the final equilibrium state. We define success as the **Center of Mass (CoM)** of the grape staying within the plate's boundary. If the CoM lies outside, the resulting instability causes the plate to tilt or the grape to fall, which is recorded as a failure.

2) **Flip Pot**
   - **Task:** Grasping the rim of a pot lying on its side and flipping it to an upright position on the table.
   - **Dataset:** 50 teleoperated demonstrations. The *pot* is initially placed within a designated workspace, where its position is perturbed with small translations (left–right and front–back) to encourage robust grasping under slight pose shifts.
   - **Evaluation:** 30 trials using the same predefined region and randomization protocol as the training set.
   - **Episode Length:** 220 timesteps (22 seconds).
   - **Success Criterion:** The pot must stand upright on its base without tipping over. This task evaluates the policy's understanding of rotational control and object pose.

Fig. 7: **Real-world robot setup.** Our experiments are conducted on a UR5e arm with an RG2 gripper. The observation is captured with a fixed front camera and a wrist-mounted camera, providing complementary viewpoints of the scene for both single-view and multi-view evaluations. All RGB images are resized to 256×256 and the robot is controlled at 10 Hz.

3) **Close Pot**
   - **Task:** Picking up the pot lid and precisely aligning it with the pot opening to close it.
   - **Dataset:** 50 teleoperated demonstrations. The *pot* is placed within a predefined region and its position is slightly jittered along the left–right and front–back axes. The *lid* is placed in a fixed, reachable starting area.
   - **Evaluation:** 30 trials where the pot's initial position is randomized within the identical workspace used for the demonstrations.
   - **Episode Length:** 160 timesteps (16 seconds).
   - **Success Criterion:** The task is considered successful if both of the following conditions are met: (1) the gripper accurately grasps the handle of the pot lid without, and (2) the lid is seated on the pot covering at least 70% of the opening area. This evaluates fine-grained gripper-object alignment and precise placement in contact-rich settings.

4) **Pick Up (Pear)**
   - **Task:** Reaching and picking up a pear placed at novel, unseen locations to test coordinate system understanding.
   - **Dataset:** 120 demonstrations collected in specific spatial clusters. We exclude areas within 5cm of the robot base or where the object is occluded.
   - **Evaluation:** 84 trials focused on unseen locations between or outside the training clusters, as shown in Fig. 5(a).
   - **Episode Length:** 50 timesteps (5 seconds).
   - **Success Criterion:** Successful grasp and lift of the pear at least 0.71m above the table surface.

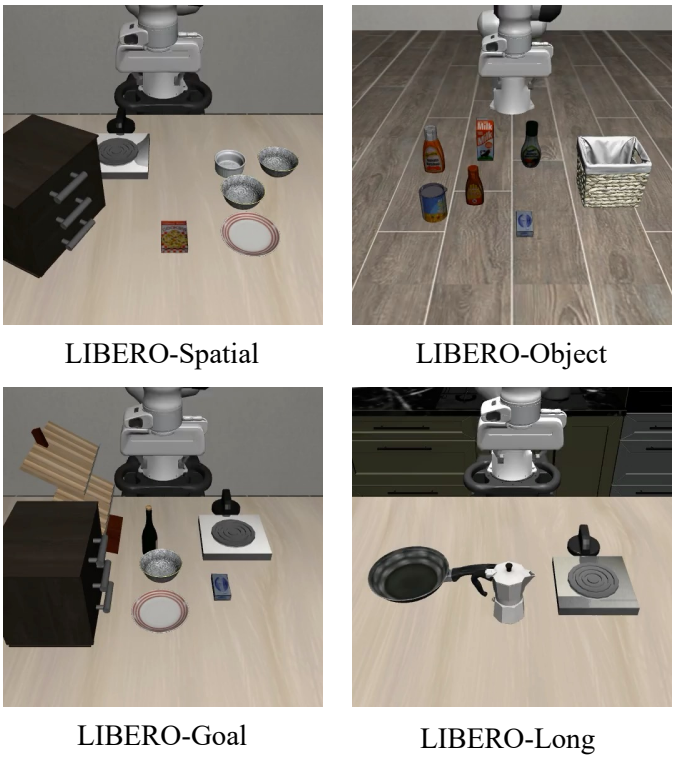


Fig. 8: **LIBERO simulation benchmark [18].** We evaluate Diffusion Policy [3] and SmolVLA [23] on LIBERO task suites, and additionally augment each policy with **AxisGuide** cues to study action coordinate grounding. We construct the *object novel position* generalization benchmark using the LIBERO Spatial suite.

B. LIBERO Simulation Benchmark

We evaluate our method in the LIBERO [18] simulation benchmark, which provides language-conditioned robot manipulation tasks. Each task is specified by an initial-state distribution and a sparse goal predicate, and an episode terminates once all goal predicates are satisfied. LIBERO is designed to systematically study knowledge transfer in lifelong learning for decision-making by disentangling shifts in declarative knowledge (objects and spatial relationships) versus procedural knowledge (motions and behaviors). LIBERO comprises four task suites that capture different types of distribution shifts; we summarize them below and refer to Fig. 8 for qualitative examples of representative tasks from each suite.

LIBERO-Spatial (10 tasks). LIBERO-Spatial is designed to isolate transfer of *spatial knowledge*. Across the 10 tasks, the manipulation objective is kept largely consistent (e.g., placing a bowl onto a plate), while the key variation is *where the target object is located* and how it is situated relative to other objects in the scene. Concretely, the suite often includes visually identical instances (e.g., two similar bowls) whose only difference is their spatial configuration; thus, solving each new task primarily requires learning and retaining new *spatial relationships* rather than new object semantics or entirely new skills.

LIBERO-Object (10 tasks). LIBERO-Object targets transfer of *object knowledge*. Here, the overall environment structure and interaction pattern are kept similar, but each task introduces a *different target object* to be manipulated (e.g., "pick up X and place it Y" with varying X). As a result, the agent must continually acquire and retain new *visual and language grounding* for novel objects while reusing largely similar low-level manipulation behaviors.

LIBERO-Goal (10 tasks). LIBERO-Goal focuses on transfer of *procedural knowledge*. In this suite, tasks share the same set of objects and maintain fixed spatial relationships, while the primary change is the *goal specification* described by the language instruction and the corresponding goal predicates. This setting reduces the need for learning new objects or new layouts and instead emphasizes learning new *behaviors and action sequences* (i.e., procedural skills) to satisfy different goals.

LIBERO-Long. To evaluate long-horizon compositional manipulation, we additionally consider the long-horizon subset of LIBERO, referred to as LIBERO-Long, which consists of tasks that require *multi-stage* behavior and longer temporal credit assignment (e.g., sequences of interactions such as opening/closing, inserting/placing, or re-orienting objects before a final placement). Compared to the 10-task suites above, LIBERO-Long places greater emphasis on *temporal composition* of skills and robustness over extended rollouts. We train and evaluate our policies on the **LIBERO-10** suites, following the standard 10-task protocol in the benchmark [18].

Demonstrations and dataset regeneration. LIBERO provides a small set of high-quality human demonstrations for each task; in the benchmark's reference setup, each task is paired with 50 teleoperated trajectories collected by human experts. In our experiments, following [14], we regenerate (replay) the released demonstration trajectories in simulation and re-render the corresponding observations under our camera configuration. We further filter the regenerated dataset by removing failed demonstrations that do not satisfy the task success condition. We use overall 1711 filtered demonstration trajectories for training.

C. Object Novel Position Experiment Details and Additional Single-View Evaluation

Experiment Details. We begin from the LIBERO-Spatial [18] suite, which is designed to test spatial knowledge transfer across tasks. LIBERO is widely used in the manipulation community and provides high-quality human demonstrations for each task, making it a convenient and reproducible starting point for constructing controlled generalization tests. However, in the standard LIBERO-Spatial setting, the target object location within each task does not vary substantially, making it difficult to directly evaluate whether a policy can *adapt its behavior* when the same object is moved to a previously unseen position. To explicitly test robustness to object relocation, we

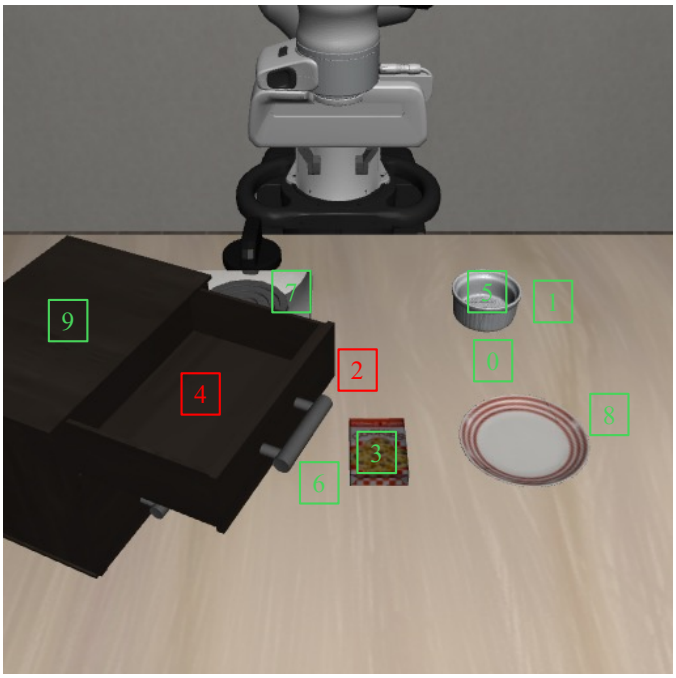
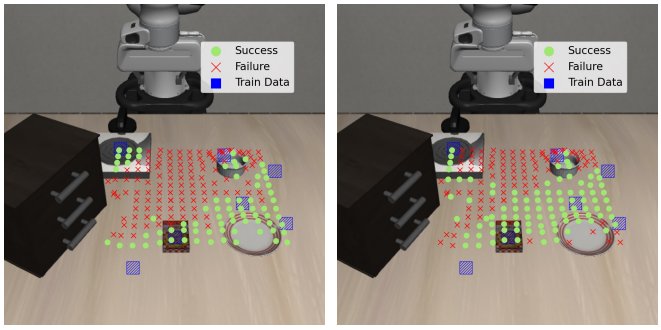
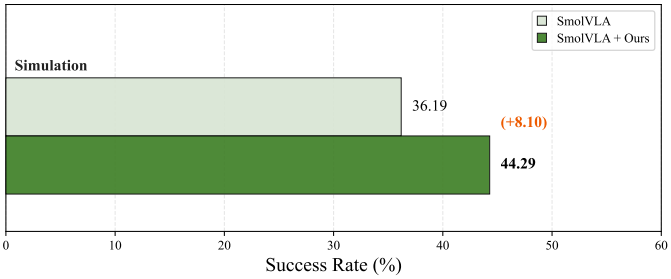


Fig. 9: **LIBERO-Spatial task setup.** Numbered boxes indicate the typical target-object region for each of the ten LIBERO-Spatial tasks (0–9), where demonstrations place the object near the corresponding region. For our *object novel position* study, we train on the remaining regions (green) while excluding tasks 2 and 4 (red). At evaluation time, we progressively expand the test placement region outward from the task-2 region to cover the full tabletop workspace, measuring generalization to unseen object locations.



SmolVLA SmolVLA + AxisGuide (Ours)

(a) Spatial Generalization Across Object Positions in Simulation



(b) Comparison of Success Rates in Simulation (Front Cam Only)

Fig. 10: **Generalization to Novel Object Positions. (Front Cam Only)** (a) shows that the baseline SmolVLA (left) generalizes poorly, with success largely confined to regions near training data, whereas SmolVLA with AxisGuide (right) reliably reaches unseen object positions between clusters, which is consistent with the multi-view results in Fig. 5. We report corresponding success rates in (b), where AxisGuide delivers substantial gains in simulation settings.

construct a modified benchmark based on LIBERO-Spatial. In LIBERO-Spatial, the target object is the *black bowl*. Since the scene contains two identical black bowls, we remove one bowl from the scene to eliminate ambiguity about which instance should be manipulated. To illustrate the spatial variation across tasks, we visualize the target black-bowl location for each LIBERO-Spatial task in Fig. 9. Based on this visualization, we exclude Task 2 and Task 4 from training (corresponding to the cases where the black bowl is placed at the table center or inside the drawer), and train policies only on the remaining 8 tasks. This modification increases the diversity of the black-bowl placement across tasks, and also introduces mild intra-task variation in the bowl position via randomized initial states. For evaluation, we perform rollouts where the episode starts with the black bowl placed at the table center, and we sweep the initial bowl position across the entire tabletop workspace.

This setting forms a new benchmark that directly tests whether the policy can visually localize the object under relocation and produce actions that correctly track and approach it under unseen spatial configurations. To measure whether the policy successfully follows the relocated object, we define success as establishing contact between the robot gripper and the black bowl (i.e., gripper-bowl contact is treated as the success condition).

Single-view Evaluation (front camera only). In the main paper, we report object relocation performance in a multi-view setting. To verify that AxisGuide also improves action understanding and object relocation robustness under a single external view, we conduct an additional experiment where the policy is trained and evaluated using only the fixed front camera. As shown in Fig 10, AxisGuide improves the success rate from 36.19% to 44.29% (+8.10%p), indicating that the benefits of AxisGuide carry over to the front-camera-only setting.

D. Viewpoint Generalizability in Single-View Training

Experiment Details. Many large-scale robot datasets contain demonstrations collected under diverse camera viewpoints, and a practical policy should remain robust when the same task is observed from novel angles. While our main paper evaluates (i) a *single-view* setup where the action coordinate semantics are fixed with respect to a static external camera, and (ii) a *multi-view* setup where a wrist-mounted camera introduces dynamically changing viewpoints, here we explicitly test whether AxisGuide remains effective when training data spans *multiple*

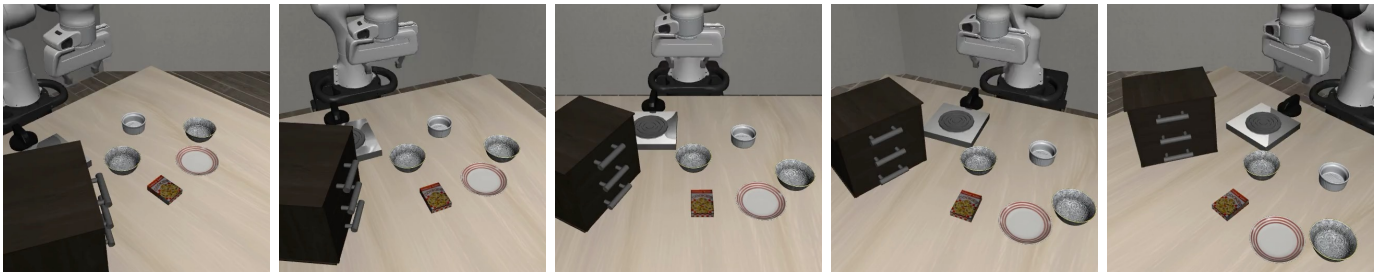


Fig. 11: **Viewpoint generalizability task setup.** Visualization of the dataset constructed by varying the camera viewpoint from $-45°$ to $45°$ in $22.5°$ increments. We use demonstrations from LIBERO-Spatial tasks 0 and 2 to train SmolVLA [23], and evaluate generalization to unseen viewpoints by testing viewpoints at $10°$ intervals.

TABLE VI: **Quantitative Comparison of Viewpoint Generalizability in the LIBERO Simulation.** We evaluate average success rates (%) over 20 rollouts.

Angle	SmolVLA [23]	SmolVLA + KYC [12]	SmolVLA + Ours
-40°	65.00	67.50	**72.50**
-30°	65.00	**82.50**	67.50
-20°	77.50	65.00	**85.00**
-10°	**82.50**	**82.50**	**82.50**
0°	72.50	**77.50**	**77.50**
10°	70.00	**82.50**	75.00
20°	82.50	87.50	**90.00**
30°	60.00	67.50	**77.50**
40°	72.50	82.50	**95.00**
Avg.	71.94	77.22	**80.28**

camera angles even under a single-view policy. To this end, we construct a controlled viewpoint generalizability benchmark in LIBERO.

Experiment. We use LIBERO task instances (task 0 and 2) and collect training demonstrations while rotating the front camera by $22.5°$ increments from $-45°$ to $+45°$. We then train policies on this multi-viewpoint dataset and evaluate them on unseen viewpoints by testing at $10°$ intervals for tasks 0 and 2. We visualize the collected viewpoints and task setup in Fig. 11. To directly assess viewpoint invariance, we use *only the front camera* as the policy input in both training and evaluation.

Results. As shown in Table VI, AxisGuide achieves the best success rates on most evaluation angles (all except $-30°$ and $+10°$). On average across viewpoints, AxisGuide improves performance by **+8.34**%p over vanilla SmolVLA and by **+3.04**%p over the viewpoint-robust baseline KYC [12]. These results indicate that AxisGuide remains effective when demonstrations are collected from diverse viewpoints and improves robustness to novel camera angles at test time.

TABLE VII: **Runtime and model size overhead of AxisGuide on SmolVLA.** End-to-end latency (batch=1) includes AxisGuide cue generation and input concatenation. AxisGuide adds only **+5.42 ms** ($\approx$0.005 s) overhead.

Method	Total params (M) ↓	ΔLatency (ms) ↓
SmolVLA (Vanilla)	450.046	0.00
SmolVLA + AxisGuide (Ours)	450.636	+5.41
Overhead (Ours − Vanilla)	**+0.590** (**+0.13%**)	**+5.41**

E. Efficiency Analysis

We analyze the computational overhead introduced by AxisGuide when applied to SmolVLA. Table VII reports both the additional learnable parameters and the end-to-end latency increase (batch=1), where latency includes AxisGuide cue generation and channel-wise concatenation. All measurements are conducted under the multi-view setting using both a wrist-mounted camera and a fixed front camera.

AxisGuide adds only **0.590M** parameters on top of the **450.046M** parameters of the vanilla model, corresponding to a negligible **+0.13%** increase. More importantly, the runtime overhead is minimal: AxisGuide incurs only **+5.41 ms** additional latency per inference, i.e., approximately **0.005 s**. These results suggest that AxisGuide provides explicit action-coordinate grounding with *negligible* compute and model-size overhead, making it practical to deploy as a lightweight visual cue to existing manipulation policies.