STEERING POLICIES WITH EMERGENT REGISTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies acquire rich manipulation skills, yet deployment may require using these skills in ways not seen during training, such as manipulating new objects, correcting an ongoing reach, or avoiding a designated region. Test-time steering can provide guidance to the policy for these issues without retraining. Prior steering methods largely modify the action-generation process, such as its latent noise or flow dynamics. We instead ask whether steering can act through visual representations within the VLM backbone. Across multiple VLAs, we find that action tokens concentrate attention on visually uninformative image patches, resembling registers in Vision Transformers. We call these *action registers*. Surprisingly, masking attention to action registers changes predicted trajectories far more than masking attention to task-relevant image tokens. We therefore introduce `RegiSteer`, which applies test-time residuals to action registers to steer frozen VLAs toward desired Cartesian directions while leaving detailed manipulation to the policy. `RegiSteer` improves manipulation across generalist and specialist policies without weight updates, remains robust to noisy spatial guidance, and supports spatial safety constraints.

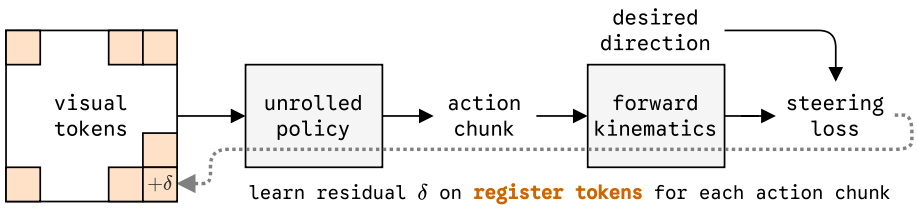


Figure 1: **`RegiSteer` steers a frozen VLA by applying a test-time residual δ to identified action registers.** A single backward pass computes a residual that locally increases end-effector progress along a desired 3D direction, while keeping all policy weights frozen.

1 INTRODUCTION

Vision-language-action (VLA) models trained on diverse robot datasets acquire manipulation skills that transfer beyond their training tasks, yet deployment conditions often differ from those seen during training. Out-of-domain objects and scenes may appear (Atreya et al., 2025), while shifts in lighting, appearance, or camera viewpoint can degrade performance (Pumacay et al., 2024). New obstacles or safety constraints may also arise only at deployment. Adapting to each such condition through retraining requires costly additional data and computation (Song et al., 2026). Instead, *inference-time steering* adapts a frozen policy using guidance available at deployment (Du & Song, 2025; Tang et al., 2026): external inputs from operators can correct ongoing motion (Chan & Kao, 2026; Kao et al., 2026), while geometric guidance can redirect the robot or steer it away from unsafe regions (Song et al., 2026).

Aforementioned steering methods largely intervene on the action-generation process, such as flow dynamics, latent noise, or reference actions. We instead ask whether a VLA can be steered from its visual representation side, before action generation. We study attention from *action tokens*, which generate robot actions, to *image tokens* within the VLM backbone. Across several VLA architectures, we observe action tokens place unusually high attention on a small set of visually

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.



Figure 2: **Register steering places more objects in challenging scenarios.** Qualitative comparison of steering methods on $\pi_{0.5}$ (Black et al., 2025a) on the RoboLab Yang et al. (2026) benchmark. Rows show `RegiSteer` (ours), inverse-kinematics on end-effector (IK-EE), Flow Control (Kao et al., 2026), and FRS (Tang et al., 2026); columns show three frames per episode. Tasks from left to right: *Put all plastic bottles away in the bin* (3 bottles), *Put hammers in the right bin* (2 hammers), and *Put all the cubes and blocks in the bin* (7 objects). All methods perform competitively on the basic pick-and-place task (right), where objects share similar geometry and grasp affordances. `RegiSteer` places more objects in the more diverse examples.

uninformative image patches, such as padding borders or static backgrounds (Fig. 3). Resembling the register phenomenon in Vision Transformers (Darcet et al., 2024), we call these image tokens *action registers*. Further investigation shows that action registers are shared across tasks, within each VLA type, appearing at fixed locations (Fig. 12). Surprisingly, masking attention to action registers changes predicted trajectories substantially more than masking attention to task-relevant image tokens (Fig. 4), suggesting uninformative visual tokens provide a compact steering interface.

Therefore, we propose `RegiSteer`, which repurposes these fixed-location action-registers as control handles by optimizing a small residual δ on them while keeping the VLA frozen. Given a desired Cartesian direction, we unroll action generation, map the predicted action chunk to end-effector motion through forward kinematics, and optimize δ to align the resulting motion with the desired direction (Fig. 1). The guidance specifies *where* the robot should move, while the pretrained VLA determines *how* to execute the manipulation and grasping from visual observations.

Across π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), and GR00T N1.6 (Bjorck et al., 2025), `RegiSteer` improves manipulation on 120-task RoboLab benchmark (Yang et al., 2026) (Figs. 2 and 6). On relational tasks, $\pi_{0.5}$ object placement rises from 15.8% to 21.7% and task success from 3.4% to 6.5%, relative gains of 37% and 91%. For π_0, relational-task success reaches 13.0%, compared with 4.9% for the strongest baseline in this category. On complex tasks, GR00T N1.6 object placement increases from 7.2% to 32.0%, versus 19.4% with IK-EE (Table 10). For OpenVLA (Kim et al., 2025), success on LIBERO-PRO (Liu et al., 2023; Zhou et al., 2025) swapped object locations rises from 0% to 65% (Table 2). After steering with `RegiSteer`, policy retains visual feedback under noisy waypoint guidance (Table 3 and Fig. 9). Importantly, steering with `RegiSteer` also enables spatial safety constraints, such as avoiding designated regions (Fig. 10).

2 Related Work

Registers and attention artifacts. Vision Transformers can develop high-norm tokens at fixed, low-information image locations that persist across inputs, receive disproportionate attention, and appear to serve as internal computational storage. Darcet et al. (2024) termed these artifacts *registers* and showed that adding dedicated register tokens during training removes them from spatial patch tokens. Jiang et al. (2025) further localized this phenomenon to a sparse set of neurons and redirected their activations into untrained register tokens at test time. Similar high-magnitude, semantically weak activations have been observed in language models (Sun et al., 2024). We observe a related phenomenon in VLAs: the same visually uninformative token locations consistently receive high action attention across tasks and environments. Unlike traditional ViT registers, however, these tokens do not exhibit unusually large value-vector norms; they are attention outliers rather than magnitude outliers. We show that these *action registers* strongly influence predicted trajectories and can serve as control handles for action generation.

Inference-time steering of robot policies. Recent work steers frozen diffusion and flow policies by modifying their action-generation process. DynaGuide uses gradients from an external dynamics model (Du & Song, 2025), while DSRL optimizes the policy's latent noise space with reinforcement learning (Wagenmaker et al., 2025). Flow Reversal Steering (FRS) maps coarse reference actions to nearby actions supported by the policy prior (Tang et al., 2026), and Flow Control (FC) modifies the initial condition of the flow process using control inputs (Kao et al., 2026). VLS synthesizes differentiable rewards with a VLM to guide diffusion or flow sampling (Liu et al., 2026). Relatedly, SkillMimicGen composes learned skills with motion-planned transit motions, providing coarse geometric guidance between skills (Garrett et al., 2025). Token Steering intervenes directly in discrete action-token space for autoregressive VLAs (Chan & Kao, 2026). Residual Semantic Steering (RSS) combines instruction expansion with a decoding rule that separates language guidance from visual affordances (Zhan et al., 2026), while FOREWARN uses a world model and a latent-aligned VLM to evaluate candidate action plans (Wu et al., 2025). `RegiSteer` instead optimizes a directional objective through visual tokens, retaining frozen weights and the original decoder.

Internal representations in VLAs. A complementary line of work studies what control-relevant information is encoded inside pretrained models. Activation steering can manipulate internal representations of multimodal models to control their behavior (Sivakumar et al., 2025). In VLAs, Häon et al. (2025) identify semantic directions such as motion direction and speed and show that intervening on them causally changes robot behavior. Other work identifies task-relevant attention heads for efficient adaptation (Mitra et al., 2025), navigation heads that detect trajectory deviations (Jeong et al., 2026), and action-conditioned attention that reveals task-relevant visual tokens (Luo et al., 2026). More general conditioning methods steer frozen transformers through learned input tokens or residual representations (Sohn et al., 2023; Das et al., 2023). We identify existing action registers through attention and causal masking, then optimize residuals on these tokens.

3 PRELIMINARIES

Actions in VLA policies. A VLA uses a vision-language model (VLM) to map images and language instructions ℓ to robot actions. At rollout timestep t, the observation $\boldsymbol{o}_t$ contains images $\boldsymbol{I}_t$ and proprioception $\boldsymbol{q}_t$. We separately denote the visual encoder E from the multimodal backbone ϕ:

$$\boldsymbol{h}_t = \phi(E(\boldsymbol{I}_t), \ell; \boldsymbol{q}_t). \tag{1}$$

The representation $\boldsymbol{h}_t$ is mapped to actions either autoregressively through discrete action tokens (Kim et al., 2025), or through a continuous diffusion/flow action head (Chi et al., 2023; Black et al., 2025a). In the latter, the action head ψ reuses $\boldsymbol{h}_t$ across N denoising or flow steps,

$$\boldsymbol{z}_{n-1} = \psi(\boldsymbol{h}_t, \boldsymbol{q}_t, \boldsymbol{z}_n, n), \qquad n = 1, \dots, N. \tag{2}$$

Here $\boldsymbol{z}_n \in \mathbb{R}^{H \times k}$ is the intermediate action chunk, refined from noise $\boldsymbol{z}_N$ to actions $\boldsymbol{a}_{t:t+H-1} = \boldsymbol{z}_0$. The chunk contains H actions of dimension k, executed by the robot's low-level PID controller.

Forward and inverse kinematics. Let $\boldsymbol{q} \in \mathbb{R}^{n_q}$ denote the arm joint configuration, excluding gripper actuation and other non-arm action dimensions. Forward kinematics (FK) maps $\boldsymbol{q}$ to the Cartesian position of the end effector (EE),

$$\boldsymbol{p}_{\mathrm{EE}} = f_{\mathrm{EE}}(\boldsymbol{q}) \in \mathbb{R}^3. \tag{3}$$

For a desired EE position $\boldsymbol{p}^{\star}_{\mathrm{EE}}$, the desired Cartesian displacement is $\Delta\boldsymbol{p}_{\mathrm{EE}} = \boldsymbol{p}^{\star}_{\mathrm{EE}} - f_{\mathrm{EE}}(\boldsymbol{q})$. Locally, FK can be linearized as

$$\Delta\boldsymbol{p}_{\mathrm{EE}} \approx \boldsymbol{J}_{\mathrm{EE}}(\boldsymbol{q})\Delta\boldsymbol{q}, \qquad \boldsymbol{J}_{\mathrm{EE}}(\boldsymbol{q}) = \frac{\partial f_{\mathrm{EE}}}{\partial \boldsymbol{q}}(\boldsymbol{q}) \in \mathbb{R}^{3 \times n_q}. \tag{4}$$

Using the pseudoinverse of the Jacobian gives the approximate joint displacement $\Delta\boldsymbol{q} = \boldsymbol{J}^{\dagger}_{\mathrm{EE}}(\boldsymbol{q})\Delta\boldsymbol{p}_{\mathrm{EE}}$. At timestep t, we therefore define the local IK-EE joint target as

$$\boldsymbol{q}^{\mathrm{IK}}_t = \boldsymbol{q}_t + \Delta\boldsymbol{q}. \tag{5}$$

IK-EE is convenient for waypoint steering because it requires only a 3D EE target rather than targets for the full arm. Cartesian-target pipelines are commonly combined with predefined grasps, IK, and collision-free motion planning (Fontanals et al., 2014; Zimmermann et al., 2020). We therefore take IK-EE as a baseline for free-space steering and compare it with `RegiSteer`, which retains the VLA's learned whole-arm behavior and is robust to noisy waypoints (Table 3 and Fig. 9).

4 METHODOLOGY

4.1 IDENTIFYING ACTION REGISTERS AND STEERING

We explain how we identify image tokens that receive concentrated attention from action queries (i.e. our action registers), test their influence by masking, and use them for steering.

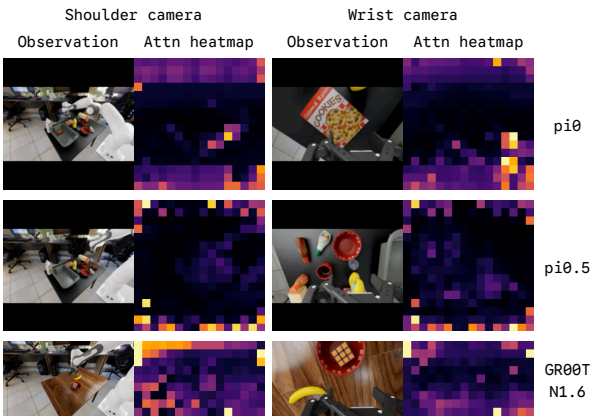


Figure 3: **Action attention concentrates on visually uninformative image regions.** Shoulder and wrist views for π_0, $\pi_{0.5}$, and GR00T N1.6 show cross-attention A_p from action queries to image patches. Peaks fall on padding borders in π_0 and $\pi_{0.5}$, and static regions in GR00T N1.6, rather than objects of interest.

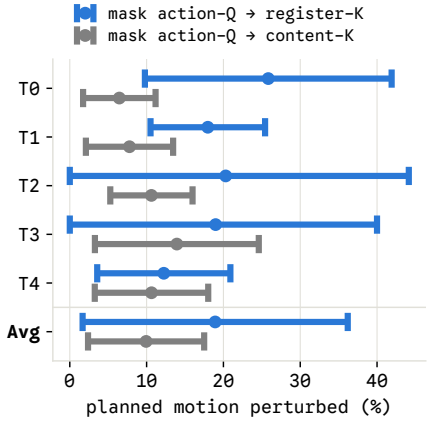


Figure 4: **Register masking perturbs trajectories more than content masking.** On RoboLab `T0-4`, masking action-query (Q) → image-key (K) attention causes mean motion changes of $\sim 29\%$ for action-register attentions and $\sim 8\%$ for action-content attentions.

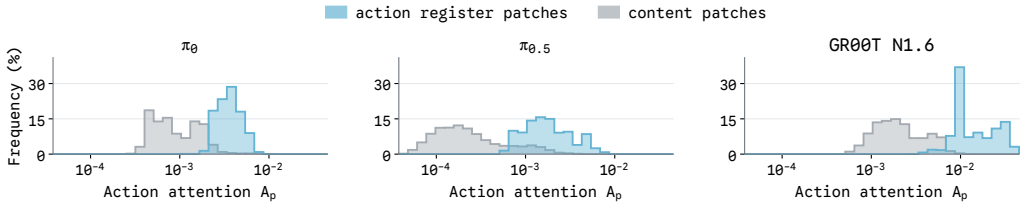


Figure 5: **Register patches located in Fig. 12 receive higher attention per patch than content patches.** Distributions of A_p on RoboLab `CondimentsInBin` for three policies.

What are the characteristics of register-like tokens in VLAs? Following the register phenomenon in ViTs (Darcet et al., 2024), we look for image tokens that receive unusually high attention from action tokens despite lying in visually uninformative regions. Across these three policies, for each image patch p, we measure the cross-attention $\alpha_{a\to p}$ it receives from the action-token queries $\mathcal{A}$, averaged over all heads N_h, all transformer layers L, and all denoising steps N, as A_p:

$$A_p = \frac{1}{|\mathcal{A}|\, N_h\, L\, N} \sum_{a\in\mathcal{A}} \sum_{h=1}^{N_h} \sum_{l=1}^{L} \sum_{n=1}^{N} \alpha_{a\to p}^{(h,l,n)}. \tag{6}$$

The heatmaps in Fig. 3 reveal high-attention, low-information patches $\mathcal{R}$, which we call candidate action registers; the remaining image tokens form the content set $\mathcal{C}$. In $\pi_{0.5}$, registers receive 60% of visual attention from action queries and account for 99.7% of the top-1% most-attended patches (Fig. 5). Unlike ViT registers, their value-vector norms are comparable to ordinary image tokens. We therefore identify registers by attention concentration and **keep their locations fixed for all visual input, even across different tasks**. Section A.1.1 in Appendix verifies the fixed selection and recurrence of action tokens $\mathcal{R}$ across tasks, camera views, and provides further ablations.

Do action registers affect the planned motion? High attention or norm alone may not establish a token's influence on the policy's output. Therefore, we test the registers by masking action-

query → image-key attention in the frozen $\pi_{0.5}$ policy. We set the corresponding pre-softmax logits to $-\infty$ across all layers and denoising steps, comparing register tokens $\mathcal{R}$ with content tokens $\mathcal{C}$. We measure the relative ℓ_2 change between masked and unmasked action chunks after mapping them to 3D EE trajectories with FK. Despite $|\mathcal{R}| \ll |\mathcal{C}|$, register masking perturbs motion by $\sim 29\%$ across five RoboLab tasks, versus $\sim 8\%$ for content masking (Fig. 4). We also perform count-matched comparisons in Fig. 17, showing a similar trend of action registers steering more effectively.

Table 1: **`RegiSteer` steering achieves the most placements with low transport jerk.** Steering channels for frozen $\pi_{0.5}$, evaluated on an empty table (single chunk) and RoboLab `CondimentsInBin` (multiple chunks), with 10 shared seeds. **Gain**: commanded-direction EE displacement per chunk, normalized by unsteered drift; negative values indicate opposite motion. **Alignment**: cosine similarity of commanded and predicted motion. **Jerk (EE/Rest)**: mean jerk of the EE and remaining arm joints during transport (m/s^3). **Placed**: objects placed per episode. Gain/alignment are omitted for the base policy; — denotes no successful grasp for jerk.

	Single-chunk, empty table		Multi-chunk, cluttered table		
Control channel	Gain ↑	Alignment ↑	Jerk (EE) ↓	Jerk (Rest) ↓	Placed ↑
Base (no steering)	—	—	6.24 ±1.28	5.74 ±2.36	0.60 ±0.60
Concat δ to inputs (Sohn et al., 2023)	7.62 ±23.3	0.140 ±0.610	—	—	0
Residual δ to content (Das et al., 2023)	1.46 ±2.6	0.620 ±0.340	6.27 ±0.48	5.91 ±0.72	0.20 ±0.20
DynaGuide (Du & Song, 2025)	2.79 ±4.04	0.332 ±0.508	49.82 ±12.45	45.31 ±12.97	0
DSRL (Wagenmaker et al., 2025)	5.81 ±4.03	0.892 ±0.139	13.63 ±11.70	8.86 ±6.91	0
FRS (analytic) (Tang et al., 2026)	3.43 ±16.3	0.086 ±0.575	—	—	0
FRS (nudge) (Tang et al., 2026)	2.08 ±1.08	0.971 ±0.027	5.71 ±1.00	5.33 ±1.28	0.60 ±0.60
ζ-controller (Buurmeijer et al., 2026)	0.46 ±0.85	0.310 ±0.504	4.77 ±2.16	3.77 ±1.03	0.20 ±0.20
OmniGuide (Song et al., 2026)	4.23 ±4.07	0.635 ±0.387	23.69 ±10.49	22.88 ±7.00	0.50 ±0.50
Flow Control (Kao et al., 2026)	3.31 ±2.41	0.815 ±0.242	6.61 ±1.92	6.50 ±2.65	0.80 ±0.49
Add δ **on registers (`RegiSteer`)**	7.16 ±8.0	0.880 ±0.210	4.76 ±1.09	4.77 ±0.44	2.46 ±0.50
IK-EE	30.69	0.920	6.11 ±2.76	7.67 ±2.30	2.08 ±0.73

Why not steer through other channels? We compare register offsets against alternative steering channels in Table 1, evaluated on a challenging `CondimentsInBin` task from RoboLab. All methods receive identical waypoints and use the same frozen $\pi_{0.5}$ policy, differing only in how the guidance is imposed. `RegiSteer` achieves strong steering with low jerk and the highest object count.

Among general-purpose methods, prompt tuning (Sohn et al., 2023) concatenates new tokens, inducing large but inconsistent motions, while residual prompting (Das et al., 2023) applies a shared residual to content tokens $\mathcal{C}$ and steers more weakly, consistent with Fig. 4. Among robotics-specific methods, DynaGuide (Du & Song, 2025), with its dynamic model retrained on RoboLab, captures dominant arm motion but misses subtle object interactions, leading to jerky transport. DSRL (Wagenmaker et al., 2025) optimizes the initial diffusion noise, providing strong control but often changing the overall behavior rather than locally correcting it, which increases jerk and manipulation failures. FRS (Tang et al., 2026) reverses a reference action through the flow: our *analytic* variant uses a Cartesian-servo reference which does not work, whereas smaller perturbations of the policy's own actions in the *nudge* variant steer more weakly. The ζ-controller (Buurmeijer et al., 2026) minimizes activation perturbations, preserving smoothness but limiting steering strength. OmniGuide (Song et al., 2026) steers strongly but with high jerk, often dropping objects. Flow Control (Kao et al., 2026) modifies the initial flow condition, but its influence fades after the policy commits, limiting repeated corrections during transport. Finally, IK-EE directly converts Cartesian guidance to arm motion, achieving strong alignment and the largest gain, but bypasses the VLA's free-space coordination and incurs higher jerk. In comparison, `RegiSteer` consequently places more objects with low jerk in comparison. We therefore utilize IK-EE, FRS, and Flow Control, the three strongest baselines by placement count, for the broader RoboLab comparison (Fig. 6).

4.2 REGISTEER

At rollout timestep t, `RegiSteer` takes observation $\boldsymbol{o}_t$, instruction ℓ, and target waypoint $\boldsymbol{p}_j^\star$ to produce a steered action chunk $\boldsymbol{a}_{t:t+H-1}$ with the policy frozen (Fig. 1). We next describe how we obtain the waypoint sequences $\boldsymbol{p}_j^\star$ and optimize the register residual δ.

From waypoints to steering directions. Sparse spatial guidance, such as waypoints and trajectory sketches, has been widely used to guide robot policies and can be obtained from perception systems or higher-level foundation models (Gu et al., 2024; Yuan et al., 2025; Tang et al., 2025; Abdolmaleki et al., 2025; Tang et al., 2026). `RegiSteer` requires only sparse 3D waypoints for free-space motion, leaving grasping, release, and detailed arm motion to the VLA, and remains robust to waypoint noise (Table 3). Since the VLAs already operate on multi-view observations, these waypoints can be obtained using standard multi-view geometry. In our experiments, we triangulate object locations from multi-view masks and generate a sparse waypoint sequence $\{\boldsymbol{p}_j^\star\}$ once at the start of each task. We explain waypoint-generation details further in Section A.4.

At rollout timestep t, the current waypoint $\boldsymbol{p}_j^\star$ and joint configuration $\boldsymbol{q}_t$ define the direction $d_t^\star$:

$$d_t^\star = \frac{\boldsymbol{p}_j^\star - f_{\mathrm{EE}}(\boldsymbol{q}_t)}{\left\|\boldsymbol{p}_j^\star - f_{\mathrm{EE}}(\boldsymbol{q}_t)\right\|}. \tag{7}$$

While rollout timestep t advances during task completion, the waypoint index j remains fixed until $\boldsymbol{p}_j^\star$ is reached; we then increment j and steer toward the next waypoint. Near the target object, we stop steering and rely on the VLA's learned grasping prior.

Steering through action registers. At timestep t, the vision encoder E produces S tokens for each of V camera views, $\boldsymbol{e}_t = E(\boldsymbol{I}_t) \in \mathbb{R}^{V\times S\times d}$. To the outputs of vision encoder $\boldsymbol{e}_t$, we add a residual $\delta \in \mathbb{R}^{V\times d}$ at the action registers $\mathcal{R}$ identified in Section 4.1:

$$\tilde{\boldsymbol{e}}_{t,s} = \begin{cases} \boldsymbol{e}_{t,s} + \delta^{(v(s))}, & s \in \mathcal{R}, \\ \boldsymbol{e}_{t,s}, & s \notin \mathcal{R}, \end{cases} \tag{8}$$

Here s indexes the VS image tokens and $v(s)$ denotes the camera view; registers in each view share a residual. The VLM maps these tokens to $\tilde{\boldsymbol{h}}_t = \phi(\tilde{\boldsymbol{e}}_t, \ell; \boldsymbol{q}_t)$ as in Eq. (1), shared by all N denoising or flow steps, which produce the action chunk $\boldsymbol{a}_{t:t+H-1}(\delta)$ as in Eq. (2).

We approximate the chunk's EE displacement as $\Delta\boldsymbol{p}_t(\delta) \approx \boldsymbol{J}_{\mathrm{EE}}(\boldsymbol{q}_t)\boldsymbol{S}(\boldsymbol{a}_{t+H-1}(\delta) - \boldsymbol{a}_t(\delta))$, where $\boldsymbol{J}_{\mathrm{EE}}(\boldsymbol{q}_t)$ is the EE Jacobian and $\boldsymbol{S} \in \mathbb{R}^{n_q\times k}$ is a fixed selection-and-scaling matrix, which extracts the arm joints (excluding the gripper) and converts normalized action differences to joint displacements. Our steering loss is the negative progress along the desired direction, $\mathcal{L}_{\mathrm{steer}}(\delta) = -\langle d_t^\star, \Delta\boldsymbol{p}_t(\delta)\rangle$.

A single-step steering update follows the normalized negative gradient,

$$\boldsymbol{g}_t = -\nabla_\delta \mathcal{L}_{\mathrm{steer}}(\delta)|_{\delta=\mathbf{0}}, \qquad \delta_t = \frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|_F}. \tag{9}$$

To compute this gradient, we unroll the N denoising or flow steps in Eq. (2) and backpropagate $\mathcal{L}_{\mathrm{steer}}$ through the generated action chunk to δ. A single backward pass gives δ_t, which we apply to generate the steered action chunk and recompute at the next policy query. If the gradient vanishes, we set $\delta_t = 0$. All policy weights remain frozen.

5 Results

We evaluate `RegiSteer` on generalist and specialist policies, test noisy guidance and safety constraints, and demonstrate physical-robot steering. Additional ablations and quantitative results appear in the Appendix.

5.1 Simulation results

Generalist policies. We evaluate FRS, Flow Control, IK-EE, and `RegiSteer` on 120 RoboLab tasks (Yang et al., 2026), using $\pi_{0.5}$, π_0, and GR00T N1.6 on the DROID embodiment (Khazatsky et al., 2024). `RegiSteer` improves object placement across all three policies (Figs. 2 and 6). For $\pi_{0.5}$, relational object placement rises from 15.8% to 21.7% and success from 3.4% to 6.5%. For π_0, relational placement rises from 1.8% to 16.1%. GR00T N1.6 reaches 32.0% placement on complex tasks, versus 7.2% for its base policy (Table 10). Across the 18 policy–category cells of Table 10, `RegiSteer` attains the highest OBJ% in 17 and ties Flow Control in the remaining one. SR%

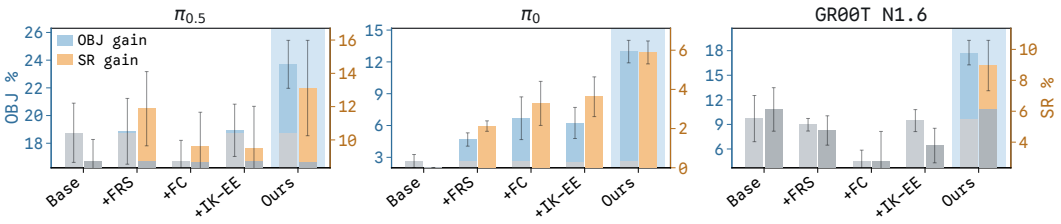


Figure 6: **Register steering improves object placement and task success across all three generalist policies.** Results on 120 RoboLab tasks (Yang et al., 2026), with gains over the base model highlighted. OBJ% measures target-object placement, while SR% requires task completion, often involving multiple objects. Methods share six seeds per task. Error bars propagate category-level variation (Table 10).

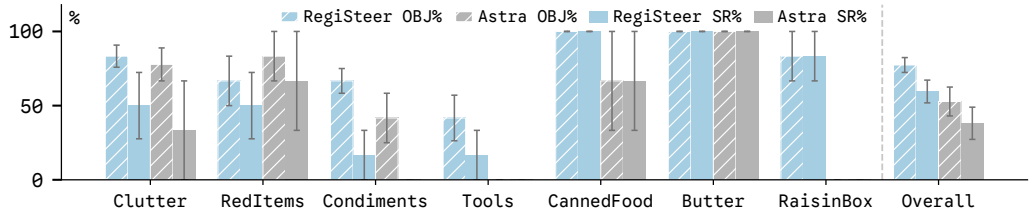


Figure 7: **RegiSteer places more objects and completes more tasks than Astra, as an end-to-end policy (high reasoning).** On seven RoboLab tasks (Yang et al., 2026), RegiSteer reaches 72.6% OBJ / 59.5% SR versus Astra's 52.4% / 38.1%. Error bars show standard error across six and three trials per task, respectively. Task-wise results appear in Section A.9.

gains are less uniform, as SR requires completing every placement in a task. Category results and qualitative comparisons appear in Sections A.6 and A.7.

In addition, we compare $\pi_{0.5}$+RegiSteer with GPT-6-Astra (high reasoning) (OpenAI, 2026) as an agentic end-to-end policy on seven challenging RoboLab tasks, using six trials per task for RegiSteer and three for Astra. RegiSteer leads by 20.2 percentage points in object placement and 21.4 in success rate (Fig. 7). We provide further details and task names in Section A.9.

Specialist policies. We test suite-finetuned OpenVLA (Kim et al., 2025) on LIBERO (Liu et al., 2023). Because RegiSteer acts before action decoding, it also applies to this autoregressive (AR) policy. OpenVLA predicts discrete action-bin probabilities and executes the highest-probability bin. We use their probability-weighted average to compute steering gradients through this non-differentiable selection, retaining the discrete decoder for execution.

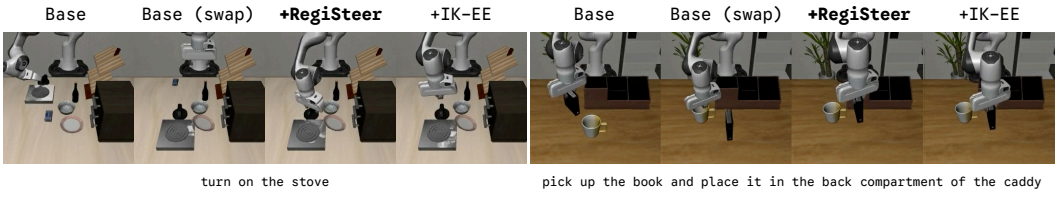


Figure 8: **RegiSteer redirects OpenVLA while preserving learned interactions.** Original LIBERO and LIBERO-PRO swap tasks. Relocating the stove knob and book exposes the base policy's bias toward training-time locations.

LIBERO-PRO swap (Zhou et al., 2025) relocates objects, reducing base-policy success from 72.5% to 0%. RegiSteer recovers 65% with frozen weights (Table 2), redirecting OpenVLA while preserving interactions such as grasping and knob rotation (Fig. 8). It also exceeds IK-EE overall (65.0% vs. 60.0%), most clearly on Goal and LIBERO-10, while IK-EE is higher on Spatial. Additional comparisons appear in Section A.8.

Table 2: **Register steering recovers 65% success after object relocation.** SR% for suite-finetuned OpenVLA (Kim et al., 2025) on LIBERO (Liu et al., 2023) and LIBERO-PRO `swap` (Zhou et al., 2025). Methods share checkpoints and five seeds per task, differing only in free-space control. FRS and Flow Control require diffusion/flow action generation and cannot use the AR decoder.

	LIBERO	LIBERO+swap (LIBERO-PRO)				
Suite	Base	Base	RegiSteer	IK-EE	FRS	Flow Control
Object	70.0 ± 6.5	0.0 ± 0.0	**85.0 ± 8.0**	80.0 ± 8.9	N/A	N/A
Spatial	78.0 ± 5.9	0.0 ± 0.0	85.0 ± 8.0	**90.0 ± 6.7**	N/A	N/A
Goal	78.0 ± 5.9	0.0 ± 0.0	**50.0 ± 11.2**	35.0 ± 10.7	N/A	N/A
10	64.0 ± 6.8	0.0 ± 0.0	**40.0 ± 11.0**	35.0 ± 10.7	N/A	N/A
Overall	72.5 ± 3.1	0.0 ± 0.0	**65.0 ± 4.8**	60.0 ± 4.7	N/A	N/A

Table 3: **Register steering tolerates waypoint noise better than IK-EE.** Objects placed at localization noise σ on three cluttered RoboLab tasks, using 10 shared seeds per task.

	BlocksInBin		CubesAndBlocksInBin		CondimentsInBin	
σ (cm)	RegiSteer	IK-EE	RegiSteer	IK-EE	RegiSteer	IK-EE
0	**3.30 ± 0.64**	3.00 ± 1.34	4.90 ± 0.54	**5.60 ± 1.50**	2.20 ± 0.60	2.20 ± 0.75
8	**1.70 ± 0.64**	1.20 ± 0.75	**2.70 ± 1.19**	2.00 ± 1.55	**0.40 ± 0.40**	0.00 ± 0.00
16	**0.70 ± 0.64**	0.30 ± 0.30	**0.90 ± 0.70**	0.80 ± 0.80	**0.30 ± 0.30**	0.00 ± 0.00

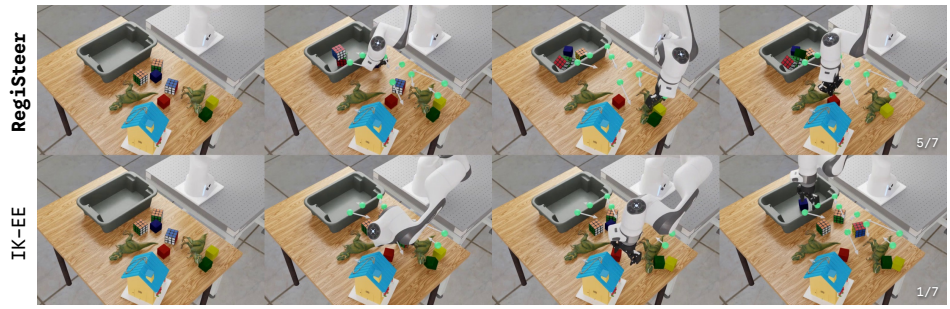


Figure 9: **RegiSteer retains visual feedback under noisy guidance.** $\pi_{0.5}$ rollouts on `CubesAndBlocksInBin` at $\sigma = 8$ cm. Green orbs mark original waypoints and white arrows their noisy offsets.

5.2 Ablation Studies

We test waypoint noise and safety constraints below. Register ablations, runtime, and linear probes appear in Sections A.1 to A.3.

Robustness to noisy waypoints. `RegiSteer` places more objects than IK-EE at both nonzero noise levels in all three tasks (Table 3 and Fig. 9). IK-EE directly tracks the noisy Cartesian target, whereas register steering only biases a policy that remains conditioned on the current images, so its visual feedback can compensate for inaccurate guidance.

Enforcing stay-away safety constraints. `RegiSteer` supports differentiable safety constraints without retraining. For spherical keep-out regions with centers $\mathbf{c}_k$ and radii r_k, we add $\mathcal{L}_{\text{repel}} = \sum_{j,k} \text{ReLU}(r_k - \|\boldsymbol{p}_j(\delta) - \mathbf{c}_k\|)^2$ to $\mathcal{L}_{\text{steer}}$, where $\boldsymbol{p}_j(\delta)$ is a predicted EE position. This reduces steps inside the region from 8.8% to 0.9% and increases closest approach from 2.7 to 7.4 cm while preserving success in this demonstration (Fig. 10).

5.3 Real-world demonstrations

We deploy `RegiSteer` on an SO-101 arm with MolmoAct2 (Fang et al., 2026), two RGB cameras, and remote H200 inference, using the same register-identification procedure. Rather than optimizing for individual waypoints, we pre-compute six reusable residuals $\{\delta_u\}$ corresponding to Cartesian

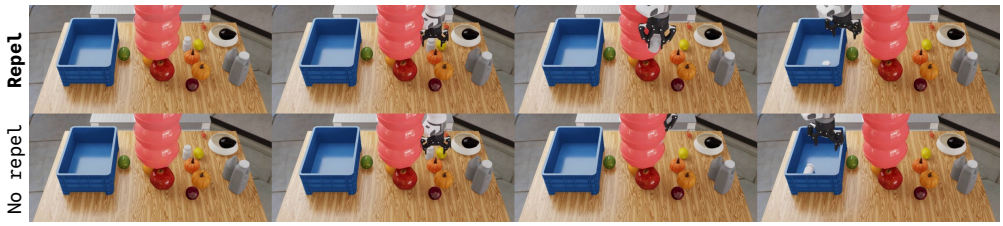


Figure 10: **Repulsion steers the EE away from the keep-out region.** On `ClutterPlasticTask` with $\pi_{0.5}$, both variants place the bottle successfully. Repulsion avoids the red region above the apple.

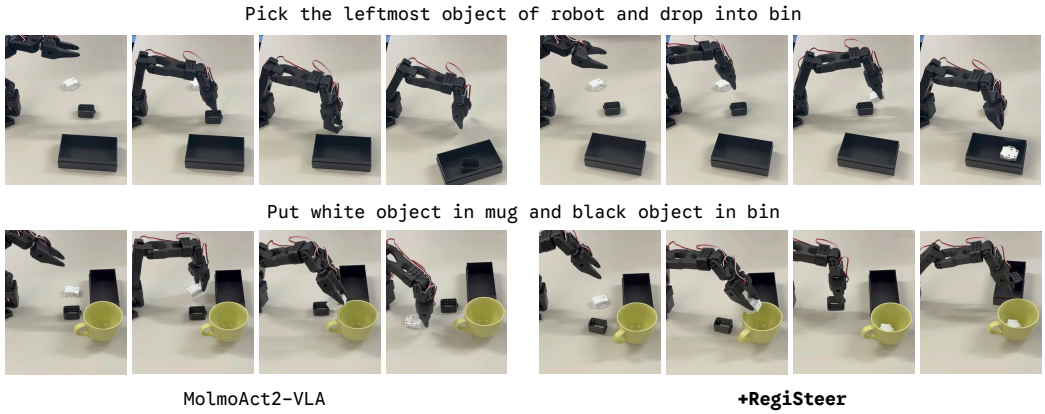


Figure 11: **Pre-computed residuals on action registers, in real-world demonstration.** VLA is MolmoAct2 (Fang et al., 2026) on SO-101 robot. Steering is along $-\hat{x}$ (top) and $+\hat{y}$ (bottom).

directions $u \in \{\pm\hat{x}, \pm\hat{y}, \pm\hat{z}\}$. Each residual is optimized offline to induce a 100 mm EE endpoint displacement in its target direction, while regularizing joint rates and deviation from the original action chunk. At deployment, steering simply selects and applies the residual for the desired direction, requiring no online optimization or backpropagation and adding 0.1% overhead.

These residuals redirect the robot while preserving MolmoAct2's manipulation behavior (Fig. 11). We apply $-\hat{x}$ steering to pick and place the leftmost object, and $+\hat{y}$ steering to place the white object in the mug and the black object in the bin. Across five held-out scenes, the six directional residuals achieve a mean alignment of 0.987, with at least 0.96 alignment in every direction and EE displacements of 50–95 mm. Details are provided in Section A.5 and Table 9.

6 Conclusion and Discussion

`RegiSteer` identifies action registers in VLAs and leverages them for test-time spatial control, while preserving learned visual and manipulation priors across generalist and specialist policies. Pre-computed directional residuals further enable steering on physical robots. Beyond steering, we hope our analyses of VLA representations provide insights toward improving future VLA policies.

Limitations. Steering remains bounded by the base VLA's manipulation capabilities and cannot compensate for absent or unreliable grasping and interaction priors (see RoboLab failures in Section A.7). Our physical experiments were further limited by MolmoAct2's sensitivity to camera viewpoint and joint backlash in the SO-101. While these experiments demonstrate real-world applicability, broader evaluation on stable physical embodiments is needed.

Future work. Why action registers emerge remains an open question. π_0 exhibits them, whereas a closely related PaliGemma-based flow policy (Atreya et al., 2025) does not (Section A.2). This suggests that factors beyond architecture, potentially including training strategies, may govern their emergence and warrant further investigation.

REFERENCES

Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Christine Chan, Oscar Chang, London Chappellet-Volpini, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, Adrian Collister, David B. D'Ambrosio, Sudeep Dasari, Todor Davchev, Meet Kirankumar Dave, Coline Devin, Norman Di Palo, Tianli Ding, Carl Doersch, Adil Dostmohamed, Yilun Du, Debidatta Dwibedi, Sathish Thoppay Egambaram, Michael Elabd, Tom Erez, Xiaolin Fang, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Ruiqi Gao, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Oliver Groth, Agrim Gupta, Roland Hafner, Steven Hansen, Leonard Hasenclever, Sam Haves, Nicolas Heess, Brandon Hernaez, Alex Hofer, Jasmine Hsu, Lu Huang, Sandy H. Huang, Atil Iscen, Mithun George Jacob, Deepali Jain, Sally Jesmonth, Abhishek Jindal, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Matija Kecman, J. Chase Kew, Donnie Kim, Frank Kim, Junkyung Kim, Thomas Kipf, Sean Kirmani, Ksenia Konyushkova, Li Yang Ku, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Tuan Anh Le, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Guy Lever, Jacky Liang, Li-Heng Lin, Fangchen Liu, Shangbang Long, Caden Lu, Sharath Maddineni, Anirudha Majumdar, Kevis-Kokitsi Maninis, Andrew Marmon, Sergio Martinez, Assaf Hurwitz Michaely, Niko Milonopoulos, Joss Moore, Robert Moreno, Michael Neunert, Francesco Nori, Joy Ortiz, Kenneth Oslund, Carolina Parada, Emilio Parisotto, Amaris Paryag, Acorn Pooley, Thomas Power, Alessio Quaglino, Haroon Qureshi, Rajkumar Vasudeva Raju, Helen Ran, Dushyant Rao, Kanishka Rao, Isaac Reid, David Rendleman, Krista Reymann, Miguel Rivas, Francesco Romano, Yulia Rubanova, Peter Pastor Sampedro, Pannag R Sanketi, Dhruv Shah, Mohit Sharma, Kathryn Shea, Mohit Shridhar, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Rachel Sterneck, Ian Storz, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Saran Tunyasuvunakool, Jake Varley, Grace Vesom, Giulia Vezzani, Maria Bauza Villalonga, Oriol Vinyals, René Wagner, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Chengda Wu, Markus Wulfmeier, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Jimmy Yan, Sherry Yang, Skye Yang, Yuxiang Yang, Hiu Hong Yu, Wenhao Yu, Wentao Yuan, Yuan Yuan, Jingwei Zhang, Tingnan Zhang, Zhiyuan Zhang, Allan Zhou, Guangyao Zhou, and Yuxiang Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv:2510.03342*, 2025.

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Cyrus Neary, Edward S. Hu, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Tony Wang, Xingfang Yuan, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martín-Martín, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. RoboArena: Distributed real-world evaluation of generalist robot policies. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 336–364. PMLR, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3B VLM for transfer. *arXiv:2407.07726*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" 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:2503.14734*, 2025.

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, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b.

Hugo Buurmeijer, Carmen Amo Alonso, Aiden Swann, and Marco Pavone. Observing and controlling features in vision-language-action models. *arXiv:2603.05487*, 2026.

Jason Chan and Jonathan C. Kao. Steering autoregressive vision-language-action policies via action token intervention. In *Proceedings of The 10th Conference on Robot Learning*, 2026.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, pp. 2632–2652, 2024.

Rajshekhar Das, Yonatan Dukler, Avinash Ravichandran, and Ashwin Swaminathan. Learning expressive prompting with residuals for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3366–3377, 2023.

Maximilian Du and Shuran Song. DynaGuide: Steering diffusion polices with active dynamic guidance. In *Advances in Neural Information Processing Systems*, volume 38, pp. 44192–44221. Curran Associates, Inc., 2025.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action reasoning models for real-world deployment. *arXiv:2605.02881*, 2026.

Joan Fontanals, Bao-Anh Dang-Vu, Oliver Porges, Jan Rosell, and Máximo A. Roa. Integrated grasp and motion planning using independent contact regions. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 887–893, 2014.

Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. SkillMimicGen: Automated demonstration generation for efficient skill learning and deployment. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2750–2790. PMLR, 2025.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. RT-Trajectory: Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, pp. 44918–44941, 2024.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2743–2762. PMLR, 2025.

Jaehwan Jeong, Evelyn Zhu, Jinying Lin, Emmanuel Jaimes, Tuan-Anh Vu, Jungseock Joo, Sangpil Kim, and M. Khalid Jawed. Your vision-language-action model already has attention heads for path deviation detection. *arXiv:2603.13782*, 2026.

Nicholas Jiang, Amil Dravid, Alexei Efros, and Yossi Gandelsman. Vision transformers don't need trained registers. In *Advances in Neural Information Processing Systems*, volume 38, pp. 56557–56595. Curran Associates, Inc., 2025.

Jonathan C. Kao, Jason Chan, and Andy Wang. Flow control: Steering vision-language-action models with simple real-time inputs. In *Proceedings of The 10th Conference on Robot Learning*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024.

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 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025.

Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

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, pp. 44776–44791. Curran Associates, Inc., 2023.

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. VLS: Steering pretrained robot policies via vision-language models. In *Proceedings of The 10th Conference on Robot Learning*, 2026.

Yulin Luo, Hao Chen, Zhuangzhe Wu, Bowen Sui, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Qiuxuan Feng, Jiale Yu, Shuo Gu, Peng Jia, Pheng-Ann Heng, and Shanghang Zhang. Look before acting: Enhancing vision foundation representations for vision-language-action models. In *Proceedings of the 34th ACM International Conference on Multimedia*, 2026.

Chancharik Mitra, Yusen Luo, Raj Saravanan, Dantong Niu, Anirudh Pai, Jesse Thomason, Trevor Darrell, Abrar Anwar, Deva Ramanan, and Roei Herzig. Mechanistic finetuning of vision-language-action models via few-shot demonstrations. *arXiv:2511.22697*, 2025.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and

Gavriel State. Isaac lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv:2511.04831*, 2025.

OpenAI. GPT-6 Astra: A New Generation of Intelligence. `https://openai.com/index/gpt-6-astra/`, September 2026. Accessed: 2026-09-25.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. In *Robotics: Science and Systems*, 2024.

Anushka Sivakumar, Andrew Zhang, Zaber Ibn Abdul Hakim, and Chris Thomas. SteerVLM: Robust model control through lightweight activation steering for vision language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23640–23665. Association for Computational Linguistics, 2025.

Kihyuk Sohn, Huiwen Chang, José Lezama, Luisa Polania, Han Zhang, Yuan Hao, Irfan Essa, and Lu Jiang. Visual prompt tuning for generative transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19840–19851, 2023.

Yunzhou Song, Long Le, Yong-Hyun Park, Jie Wang, Junyao Shi, Lingjie Liu, Jiatao Gu, Eric Eaton, Dinesh Jayaraman, and Kostas Daniilidis. OmniGuide: Universal guidance fields for enhancing generalist robot policies. In *Proceedings of The 10th Conference on Robot Learning*, 2026.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Conference on Language Modeling*, 2024.

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering. *arXiv:2606.13675*, 2026.

Grace Tang, Swetha Rajkumar, Yifei Zhou, Homer Rich Walke, Sergey Levine, and Kuan Fang. KALIE: Fine-tuning vision-language models for open-world manipulation without robot data. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9507–9515, 2025.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 2025.

Yilin Wu, Thomas Tian, Gokul Swamy, and Andrea Bajcsy. From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment. In *Proceedings of Robotics: Science and Systems*, 2025.

Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. RoboLab: A High-Fidelity Simulation Benchmark for Analysis of Task Generalist Policies. In *Proceedings of Robotics: Science and Systems*, 2026.

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 in robotics. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4005–4020. PMLR, 2025.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986, 2023.

Zhihao Zhan, Yuhao Chen, Jiaying Zhou, Qinhan Lyu, Hao Liu, Keze Wang, Liang Lin, and Guangrun Wang. Stable language guidance for vision–language–action models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4137–4159. Association for Computational Linguistics, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv:2510.03827*, 2025.

Simon Zimmermann, Ghazal Hakimifard, Miguel Zamora, Roi Poranne, and Stelian Coros. A multi-level optimization framework for simultaneous grasping and motion planning. *IEEE Robotics and Automation Letters*, 5(2):2966–2972, 2020.

A APPENDIX

Contents **Page**

A.1 Ablations on action registers . . . 15
A.1.1 Identifying action registers across tasks . . . 16
A.1.2 Action registers under a different camera view . . . 20
A.1.3 Vector norm, attention magnitude, and number analyses . . . 21
A.1.4 Runtime analysis . . . 22
A.2 Which policies develop action registers, and why? . . . 22
A.3 Linear probing on canonical axes . . . 23
A.4 Waypoint assignment . . . 23
A.5 Real-world deployment . . . 24
A.6 RoboLab quantitative results . . . 24
A.7 RoboLab qualitative results . . . 25
A.8 LIBERO qualitative results . . . 29
A.9 Direct visual control with Astra . . . 31
A.10 AI disclosure information . . . 34

A.1 ABLATIONS ON ACTION REGISTERS

We identify candidate registers from recurring attention in visually uninformative regions: padded borders for π_0 and $\pi_{0.5}$, and background edges and corners for GR00T (Fig. 12). We inspect multiple tasks, quantify recurrence, and check another camera view before testing the selected tokens through masking. Their positions stay fixed during steering. We measure attention using Eq. (6): cross-attention for separate action and image streams, or causal self-attention for a shared sequence.

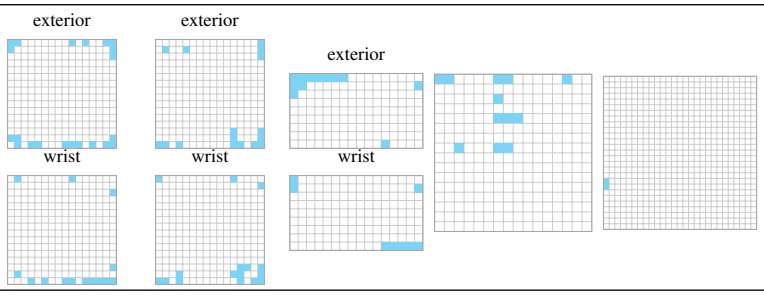
	$\pi_{0.5}$	π_0	GR00T N1.6	OpenVLA	MolmoAct2	`paligemma_diffusion`
Embodiment Type	DROID Generalist	DROID Generalist	DROID Generalist	Franka Specialist (LIBERO)	SO-101 Generalist	DROID Generalist
Action registers						N/A
Token grid	16 × 16 per view	16 × 16 per view	9 × 16 per view	16 × 16	27 × 27	16 × 16

Figure 12: **Register locations differ across policies and camera inputs. However, they are common across tasks.** For π_0, $\pi_{0.5}$, and GR00T, cyan marks locations among the top-10% attention patches in all six tasks, restricted to padding for the π models (Section A.1.1). These recurrence maps are distinct from the fixed experimental masks. Other policies show their identified masks; `paligemma_diffusion` has none.

A.1.1 IDENTIFYING ACTION REGISTERS ACROSS TASKS

We identify action registers by asking which high-attention image locations recur across tasks despite changes in scene content and robot configuration. Qualitatively, attention maps across six RoboLab tasks show a consistent pattern (Figs. 13 to 15): high action attention repeatedly concentrates at padded image borders for the π models and at background edges and corners for GR00T.

We quantify this recurrence independently for each policy and camera view using the following procedure:

1. **Aggregate attention within each task.** We average patch-level action attention over one episode for each of the six tasks.
2. **Select high-attention locations.** For each task, we retain the top 10% of image patches by attention, corresponding to 26 of 256 patches for the π models and 14 of 144 patches for GR00T.
3. **Identify shared locations across tasks.** We intersect the six task-specific sets and define locations appearing in the top 10% for every task as shared high-attention locations.

This procedure yields the locations visualized in Fig. 12: 16 exterior and 16 wrist patches for π_0, 21 exterior and 14 wrist patches for $\pi_{0.5}$, and 12 exterior and 8 wrist patches for GR00T. Notably, **every shared location falls within the visually uninformative regions identified above; none lies within scene-content regions.** Thus, the locations receiving consistently high action attention are spatially fixed across tasks rather than tracking task-relevant objects or robot configurations.

We further quantify the stability of the full top-10% sets using **pairwise intersection over union (IoU)**. For each policy and camera view, we compute IoU for all 15 pairs of the six tasks and report the mean and standard deviation in Table 4. Across-task IoU ranges from 0.613 to 0.899, far above the approximately 0.05 expected for random subsets of the same size. High action attention therefore concentrates at substantially overlapping image locations across the tested tasks.

Table 4: **High-attention locations are consistent across tasks.** Across-task IoU is the mean $\pm$ standard deviation over all 15 pairs of six tasks; random subsets of equal size have expected IoU ≈ 0.05. Shared sites are locations that appear among the top-10% patches in all six tasks and are visualized in Fig. 12.

Policy	View	Across-task IoU	Shared sites
π_0	Exterior	0.727 ± 0.055	16
	Wrist	0.819 ± 0.046	16
$\pi_{0.5}$	Exterior	0.899 ± 0.049	21
	Wrist	0.784 ± 0.071	14
GR00T N1.6	Exterior	0.879 ± 0.086	12
	Wrist	0.613 ± 0.075	8

These analyses establish that high-attention locations recur across tasks; the steering and masking experiments continue to use their original fixed register masks. We next test whether these locations also remain stable under changes in camera viewpoint (Section A.1.2).

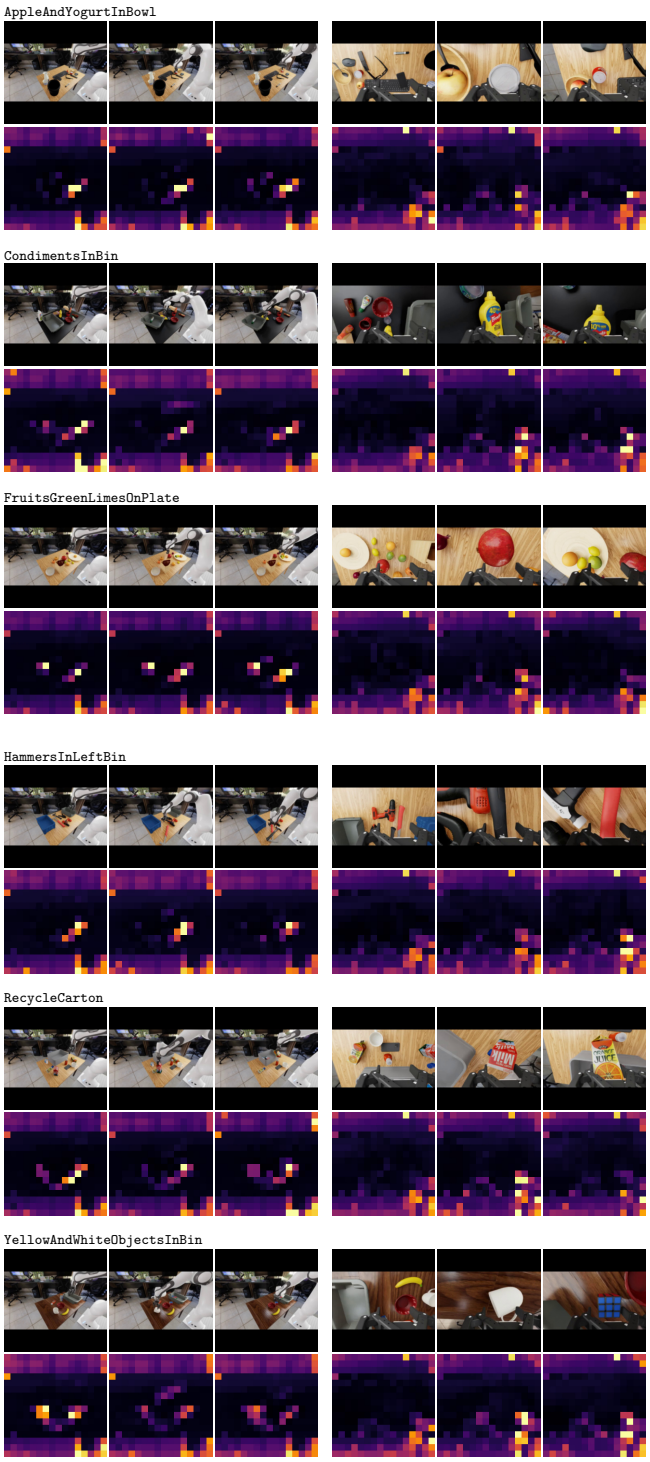


Figure 13: π_0 **retains recurring attention locations across different tasks.** Padding bands and an interior cluster coexist with changing scene content. For each task, the left and right triplets show exterior and wrist views, respectively, at the first, middle, and last sampled times. RGB images appear above their action-to-image attention maps.

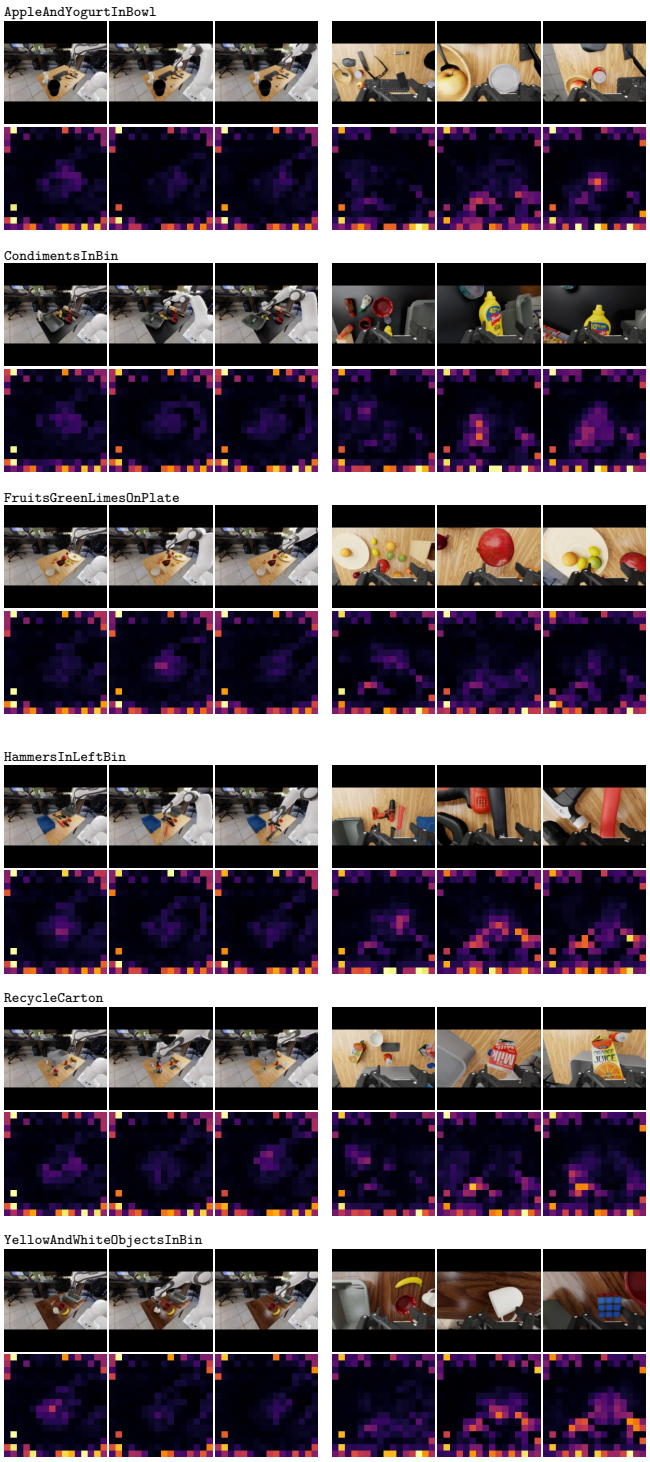


Figure 14: $\pi_{0.5}$ **maintains attention at padding locations across tasks.** High-attention patches recur along image borders as scene content changes. For each task, the left and right triplets show exterior and wrist views, respectively, at the first, middle, and last sampled times. RGB images appear above their action-to-image attention maps.

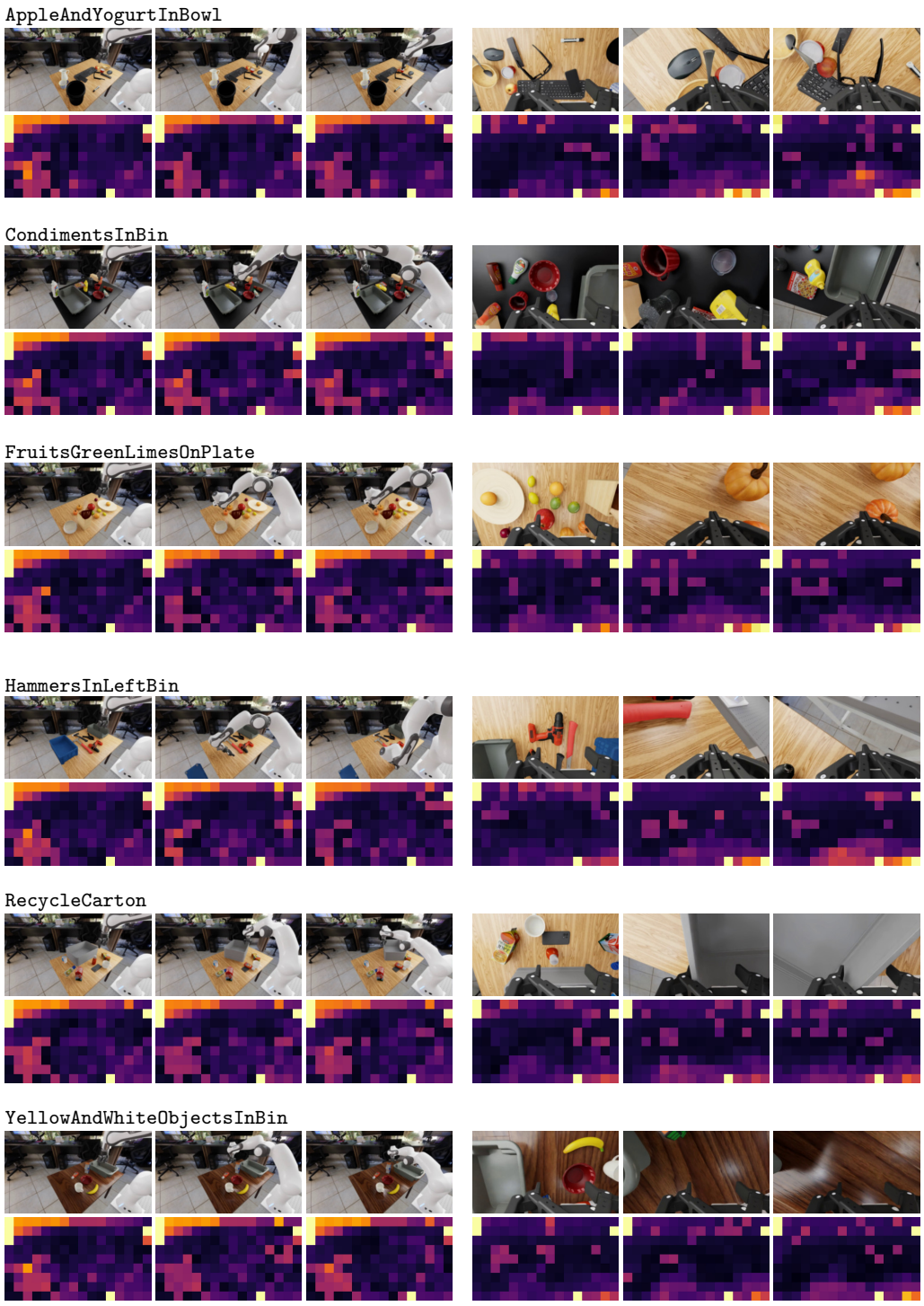


Figure 15: **GR00T N1.6 exhibits recurring corner attention across tasks.** Both views retain common high-attention locations, with more variation in the wrist view. For each task, the left and right triplets show exterior and wrist views, respectively, at the first, middle, and last sampled times. RGB images appear above their action-to-image attention maps.

A.1.2 ACTION REGISTERS UNDER A DIFFERENT CAMERA VIEW

We test whether action-register locations remain stable under a change in camera viewpoint. We replace the standard left-shoulder image with RoboLab's right-shoulder view while holding the scene, robot state, instruction, and wrist image fixed. We evaluate three tasks: `AppleAndYogurtInBowl`, `CondimentsInBin`, and `HammersInLeftBin`.

For each scene, we measure the intersection **over union (IoU) of high-attention register and content locations between the two shoulder views**, averaging attention over three matched noise draws. We select the same number of locations in each view and report the expected IoU of random selections of equal size as a reference.

Table 5: **Register attention is more spatially stable than content attention across camera views.** Overlap is the IoU of high-attention locations between left- and right-shoulder views, reported as mean ± standard deviation over three scenes. Chance denotes the expected IoU for random selections of equal size.

	Measured overlap		Chance overlap	
Policy	Registers	Content	Registers	Content
π_0	0.847 ± 0.049	0.216 ± 0.021	0.093	0.069
$\pi_{0.5}$	0.816 ± 0.093	0.171 ± 0.020	0.093	0.069
`GR00T N1.6`	1.000 ± 0.000	0.204 ± 0.065	0.311	0.021

Across all three policies, high-attention register locations remain highly consistent between shoulder views, with IoU of 0.82–1.00, compared with 0.17–0.22 for content locations (Table 5). The attention maps in Fig. 16 show the same pattern. Together, these results indicate that register locations are substantially more stable than content-attention locations under this camera-view change.

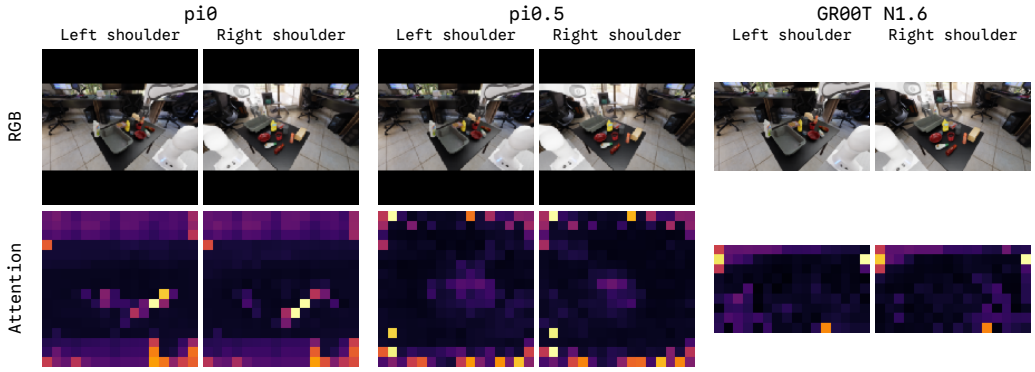


Figure 16: **High attention remains concentrated at similar register locations across shoulder views.** Paired left- and right-shoulder views of `CondimentsInBin` for π_0, $\pi_{0.5}$, and GR00T N1.6, together with their attention maps. Each model uses a shared color scale across the two views.

A.1.3 TOKEN COUNT, ATTENTION, AND VALUE-NORM ANALYSES

Controlling for the number of masked tokens. The register set $\mathcal{R}$ and content set $\mathcal{C}$ defined in Section 4.1 contain different numbers of tokens. To control for set size, we mask equal numbers of tokens randomly sampled from $\mathcal{R}$ and $\mathcal{C}$. Across tested subset sizes and policies, masking register tokens consistently perturbs predicted motion more than masking the same number of content tokens (Fig. 17).

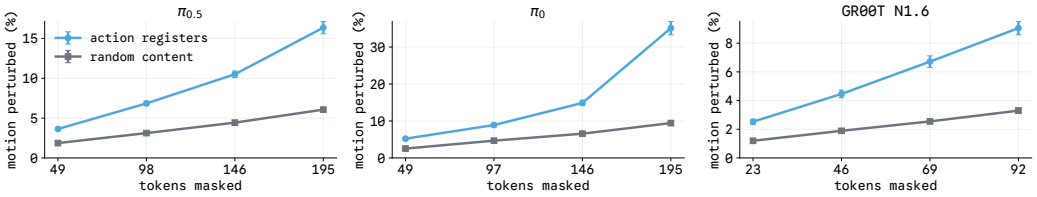


Figure 17: **Action registers affect planned motion more than an equal number of content tokens.** We mask attention from action queries to equal numbers of randomly sampled register (blue) or content (gray) tokens. Trajectory changes are normalized by the magnitude of the original planned motion. Variation reflects repeated sampling at each subset size.

Attention versus value-vector norm. Registers receive $2.21\times$–$7.86\times$ more attention than content tokens, despite their value-vector norms being only $0.62\times$–$0.87\times$ as large (Table 6 and Fig. 18). Their high attention therefore cannot be attributed to unusually large value vectors.

Table 6: **High action-register attention is not explained by larger value vectors.** Ratios between action registers and content tokens for action attention and value-vector norm.

Policy	Attention	Value norm	Top-1% action registers
$\pi_{0.5}$-DROID	$2.48\times$	$0.87\times$	99.7%
π_0-DROID	$2.21\times$	$0.77\times$	43.2%
GR00T N1.6	$\mathbf{7.86}\times$	$0.62\times$	100.0%

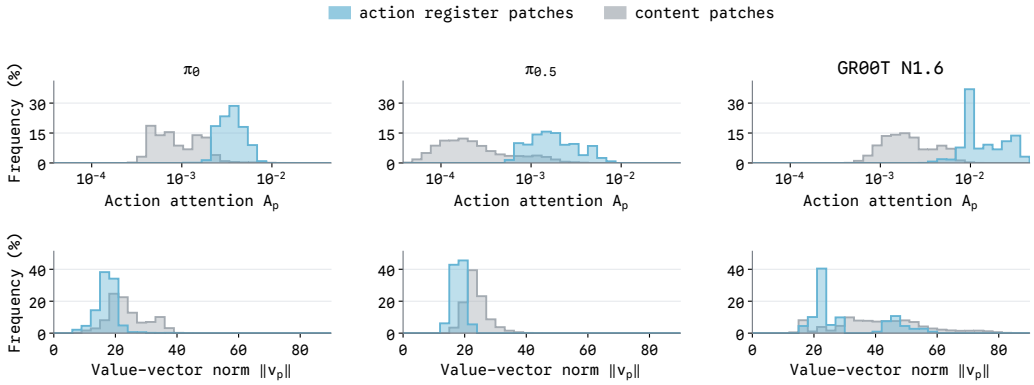


Figure 18: **Action registers are characterized by high attention, not large value-vector norms.** *Top:* Distribution of action-to-image attention A_p over register (blue) and content (gray) patches. Across all three policies, high attention concentrates on register tokens. π_0 and $\pi_{0.5}$ share a PaliGemma/SigLIP backbone and padding layout and exhibit similar distributions, whereas GR00T N1.6 uses a different backbone and concentrates attention on a smaller register set. *Bottom:* Value-vector norms $\|v_p\|$ for the same patches. Unlike classical ViT registers (Darcet et al., 2024), action registers do not exhibit unusually large norms: their norms are $0.62\times$–$0.87\times$ those of content tokens despite receiving $2.21\times$–$7.86\times$ more attention. We therefore identify action registers through attention concentration rather than feature magnitude.

A.1.4 RUNTIME ANALYSIS

All task experiments in this paper use the single-gradient (`grad`) version of `RegiSteer`. We compare its runtime with the base policy and iterative optimization (`opt`, K updates). Measurements use shared H200 GPUs, batch size one, and four flow steps. We time the complete model computation, including steering, after warmup; input preparation and robot/simulator communication are excluded.

Table 7: **Single-gradient steering is substantially faster than iterative optimization.** Model latency in milliseconds, reported as median [25th, 75th percentile] over 15 synchronized queries across three scenes per method and model.

Method	π_0 (ms)	$\pi_{0.5}$ (ms)
Base	54 [53, 55]	57 [54, 58]
`RegiSteer` (`grad`)	163 [161, 163]	177 [144, 178]
`RegiSteer` (`opt`, $K = 40$)	3084 [2936, 3228]	3379 [3119, 3547]
`RegiSteer` (`opt`, $K = 120$)	8938 [8840, 9077]	10009 [9712, 10195]

`grad` takes 163–177 ms per action chunk, about $3\times$ the base policy, whereas `opt` takes 3–10 s. The real-world demonstrations reuse offline residuals, adding only approximately 0.1% inference overhead (Section A.5).

A.2 WHICH POLICIES DEVELOP ACTION REGISTERS, AND WHY?

Action registers are not determined by architecture or image geometry alone. Policies with nearly identical architectures can differ in whether registers emerge, while policies with different backbones can both develop them, suggesting that training also plays an important role.

We first compare π_0, $\pi_{0.5}$, and `paligemma_diffusion` on DROID (Beyer et al., 2024; Black et al., 2025b;a; Atreya et al., 2025; Khazatsky et al., 2024). All three use a PaliGemma/SigLIP backbone with the same image preprocessing (Zhai et al., 2023), yielding 256 image patches, including 98 fixed padding patches. To measure whether attention concentrates on these padding locations, we compute the ratio of mean attention to padding tokens over mean attention to content tokens. This border/content ratio is 2.81 for $\pi_{0.5}$ and 2.35 for π_0, but only 0.54 for `paligemma_diffusion` (Table 8). Thus, sharing the same backbone and padding geometry is insufficient for action registers to emerge.

Table 8: **Shared architecture and padding do not ensure register formation.** Related PaliGemma-based policies differ substantially in their attention to padding tokens on the same DROID observation.

Policy	Border/content	Action registers?
$\pi_{0.5}$-DROID	**2.81**$\times$	yes
π_0-DROID	2.35$\times$	yes
`paligemma_diffusion`	0.54$\times$	no

Conversely, action registers are not specific to the PaliGemma family. GR00T N1.6 (Bjorck et al., 2025), which uses a different backbone and action head, develops registers at fixed image corners. These register tokens receive $7.86\times$ more attention than content tokens and account for all of the top-1% most-attended image tokens (Table 6 and Fig. 18).

Together, these observations suggest that register formation depends on factors beyond architecture and image geometry, potentially including the training procedure. Establishing which factors cause registers to emerge would require controlled training runs and analysis across intermediate checkpoints, which we leave to future work.

A.3 Linear probing on canonical axes

We test whether a linear probe can recover the commanded direction from the register residual $\boldsymbol{\delta}$. During $\pi_{0.5}$ rollouts on nine RoboLab tasks we log the steering solution at every online re-solve, giving 2264 records, each pairing a commanded unit end-effector direction $\boldsymbol{u} \in \mathbb{R}^3$ with its optimized residual $\boldsymbol{\delta} \in \mathbb{R}^{4096}$. Per task, we center the residuals and project them onto their top r principal components, $\boldsymbol{c} = \boldsymbol{W}^\top(\boldsymbol{\delta} - \bar{\boldsymbol{\delta}})$, then fit a linear probe

$$\hat{\boldsymbol{u}} = \boldsymbol{A}_{\text{task}}\,\boldsymbol{c}, \qquad \boldsymbol{A}_{\text{task}} \in \mathbb{R}^{3\times r}, \tag{10}$$

whose three outputs estimate the world-frame x, y, and z components of the commanded displacement. We pool the per-task predictions in their shared world frame and color points by the dominant commanded axis. Under 5-fold cross-validation, the directional separation in Fig. 19 indicates that the commanded axis is linearly decodable from the residual. We also observe a bias towards $-x$ direction because $\pi_{0.5}$ has an implicit bias of moving towards the table.

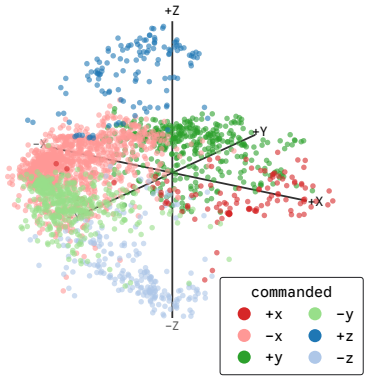


Figure 19: Illustration of directional probing of register residuals.

A.4 Waypoint assignment

Here we detail how we construct the sparse waypoint sequence $\{\boldsymbol{p}_j^\star\}$ used to define the steering direction $d_t^\star$ in Eq. (7). Waypoints are generated once at the start of each task and remain fixed throughout the rollout.

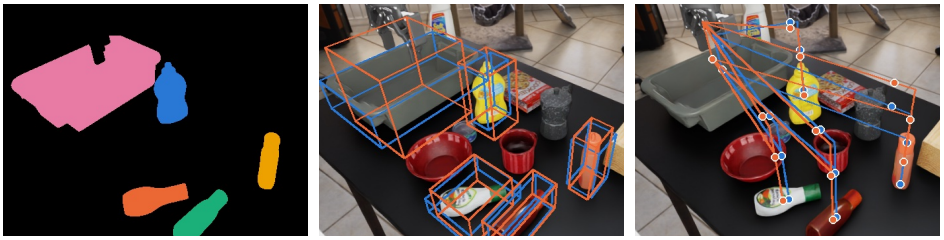


Figure 20: **Calibrated silhouettes provide coarse spatial guidance without specifying grasp points.** Waypoint-generation pipeline. Orange shows 3D bounding boxes and waypoints estimated from calibrated multi-view observations; blue shows their counterparts obtained from simulator ground-truth geometry.

We first identify the objects relevant to the task. In RoboLab, these objects are specified by the task definition, although they could also be proposed by a higher-level vision-language agent, as in Abdolmaleki et al. (2025). For each object, we obtain a binary mask in every calibrated camera view. These masks can be generated using a text-grounded segmentation model; for our large-scale RoboLab experiments, we instead use simulator-provided 2D masks (Mittal et al., 2025). From these 2D masks, we estimate object geometry using only calibrated multi-view information. We triangulate the mask centroids to initialize the 3D location, construct a local voxel grid, and apply shape-from-silhouette (Laurentini, 1994) by retaining voxels that project inside the corresponding masks across views. The surviving volume defines a coarse 3D bounding box, from which we place offset waypoint anchors and interpolate the sequence $\{\boldsymbol{p}_j^\star\}$. These multi-view estimates are shown in orange in Fig. 20. For reference, blue shows the same construction using simulator 3D geometry.

Importantly, **we never place waypoints at grasp, contact, or other object-affordance points**. The waypoints guide only free-space approach and transport. After the waypoint near the object is reached, steering is disabled and the VLA determines the grasp, release, orientation, and whole-arm motion from its learned visual and manipulation priors.

A.5 REAL-WORLD DEPLOYMENT

We deploy `RegiSteer` on an SO-101 arm with MolmoAct2 (Fang et al., 2026), two RGB cameras, and remote inference on an H200. Registers are identified once, as in simulation. Here we pre-compute six residuals $\{\delta_u\}$, one per Cartesian direction $u \in \{\pm\hat{x}, \pm\hat{y}, \pm\hat{z}\}$ from our empty table setup, instead of optimizing at each policy query.

Specifically, each residual δ_u is optimized to shift the predicted end-effector (EE) endpoint by $d = 100\,\text{mm}$ along direction u:

$$\delta_u = \arg\min_{\delta} \left\| \big(\boldsymbol{p}_T(\delta) - \boldsymbol{p}_T(0)\big) - du \right\|^2 + \lambda \|\delta\|^2. \tag{11}$$

Here, $\boldsymbol{p}_T(\delta)$ is the predicted EE endpoint, computed with differentiable FK, and $\lambda = 0.1$. We use identical flow-sampling noise for $\boldsymbol{p}_T(\delta)$ and $\boldsymbol{p}_T(0)$ to isolate the residual's effect. At deployment, we add the residual for the desired direction to the registers, with no online optimization or weight updates. This adds approximately 0.1% inference overhead, excluding remote communication. Across five held-out scenes, the six residuals achieve mean directional alignment of 0.987 (at least 0.96 per direction) and 50–95 mm EE displacement (Table 9).

Table 9: **Directional steering with pre-computed residuals.** Results across six Cartesian directions and five held-out scenes.

Directions	Scenes	Mean alignment	Min. alignment	EE displacement
6	5	0.987	≥ 0.96	50–95 mm

A.6 ROBOLAB QUANTITATIVE RESULTS

Table 10 breaks down the 120-task evaluation by difficulty and competency, using the same seeds across steering methods.

Table 10: **Register steering improves object placement over each base policy in every reported category.** Scores on the 120-task RoboLab (Yang et al., 2026) suite. OBJ% is the percentage of objects correctly picked and placed; SR% is the percentage of completed tasks, which may require multiple placements. We evaluate six seeds per task, shared across methods. Difficulty and competency columns group the same task suite.

		RoboLab (Difficulty, 120 tasks)						RoboLab (Competency, 120 tasks)					
		Simple		Moderate		Complex		Visual		Relational		Procedural	
Model	Method	OBJ%	SR%	OBJ%	SR%	OBJ%	SR%	OBJ%	SR%	OBJ%	SR%	OBJ%	SR%
$\pi_{0.5}$	Base	16.6 ±2.1	10.9 ±2.0	18.7 ±5.2	8.0 ±2.1	27.0 ±4.8	2.2 ±2.1	19.5 ±3.4	9.5 ±0.2	15.8 ±3.5	3.4 ±1.5	25.5 ±9.2	3.1 ±2.5
	+FRS	14.9 ±3.3	12.9 ±3.1	22.6 ±4.8	13.8 ±4.4	25.2 ±2.3	3.6 ±3.1	20.3 ±3.3	12.7 ±3.3	15.5 ±1.9	7.7 ±1.5	24.6 ±2.0	7.3 ±1.8
	+Flow Control	13.0 ±2.5	10.0 ±1.8	17.7 ±1.5	9.1 ±4.5	28.4 ±3.4	9.5 ±7.2	19.8 ±2.0	10.8 ±2.5	12.2 ±3.1	4.7 ±1.4	26.9 ±4.5	9.7 ±6.8
	+IK-EE	15.5 ±1.9	11.1 ±1.8	18.7 ±3.8	8.8 ±7.0	32.4 ±7.1	4.9 ±4.3	22.1 ±4.5	9.3 ±4.0	14.1 ±4.0	3.6 ±1.6	24.4 ±7.9	5.9 ±5.1
	+RegiSteer	20.4 ±1.2	15.7 ±3.1	25.1 ±4.9	11.5 ±6.5	32.9 ±1.6	7.1 ±7.1	24.4 ±2.1	13.3 ±3.8	21.7 ±2.1	6.5 ±1.4	30.5 ±2.2	6.9 ±2.4
π_0	Base	2.4 ±0.7	0.0 ±0.0	3.1 ±1.5	0.0 ±0.0	2.5 ±0.8	0.0 ±0.0	1.4 ±0.9	0.0 ±0.0	1.8 ±1.4	0.0 ±0.0	1.4 ±0.0	0.0 ±0.0
	+FRS	3.8 ±0.0	2.1 ±0.5	5.9 ±1.9	3.1 ±0.0	5.2 ±0.8	0.0 ±0.0	4.4 ±0.0	1.1 ±0.4	4.6 ±0.3	3.7 ±1.2	5.0 ±0.7	0.0 ±0.0
	+Flow Control	4.3 ±2.2	3.6 ±1.8	8.2 ±4.9	4.2 ±1.8	12.2 ±2.7	0.0 ±0.0	6.7 ±0.7	3.8 ±1.3	7.2 ±2.4	4.1 ±2.8	13.3 ±0.8	0.0 ±0.0
	+IK-EE	5.4 ±1.9	5.7 ±1.8	6.2 ±3.1	1.8 ±1.0	9.5 ±2.3	0.0 ±0.0	5.9 ±1.8	2.1 ±0.7	6.0 ±0.0	4.9 ±0.0	9.5 ±2.2	0.0 ±0.0
	+RegiSteer	12.0 ±0.1	4.7 ±0.0	14.9 ±3.2	10.4 ±1.8	12.2 ±1.4	0.0 ±0.0	10.0 ±0.0	1.3 ±0.0	16.1 ±1.4	13.0 ±1.4	14.8 ±1.6	0.0 ±0.0
GR00TN1.6	Base	11.6 ±5.1	8.3 ±2.4	7.7 ±1.5	5.2 ±1.8	7.2 ±3.9	0.0 ±0.0	5.7 ±1.9	3.4 ±3.0	10.8 ±1.2	8.1 ±1.4	9.5 ±5.0	0.0 ±0.0
	+FRS	7.8 ±1.3	7.0 ±1.6	7.7 ±0.8	2.1 ±0.9	16.2 ±1.4	2.4 ±2.1	9.8 ±1.2	5.1 ±1.7	6.4 ±1.8	3.3 ±1.9	17.1 ±1.9	2.1 ±2.1
	+Flow Control	4.3 ±1.9	5.2 ±3.6	4.1 ±2.4	0.0 ±0.0	6.3 ±2.8	0.0 ±0.0	4.7 ±0.6	3.0 ±2.0	2.8 ±1.8	2.4 ±2.4	8.1 ±3.0	0.0 ±0.0
	+IK-EE	7.6 ±1.1	5.2 ±0.9	8.2 ±3.6	3.1 ±3.1	19.4 ±2.1	0.0 ±0.0	10.6 ±1.0	3.8 ±1.3	5.2 ±2.5	1.6 ±1.4	19.0 ±0.8	2.4 ±1.4
	+RegiSteer	16.7 ±1.7	13.5 ±2.4	13.3 ±3.2	5.4 ±3.1	32.0 ±4.1	0.0 ±0.0	20.0 ±1.2	8.5 ±2.7	12.4 ±0.7	7.3 ±4.2	32.4 ±3.0	0.0 ±0.0

A.7 ADDITIONAL ROBOLAB QUALITATIVE RESULTS

Figures 21 to 28 compare `RegiSteer`, `IK-EE`, `FRS`, `Flow Control`, and the base $\pi_{0.5}$ policy on six tasks, followed by two `RegiSteer` failure cases. Each rollout shows four independently selected, chronological front-view frames.

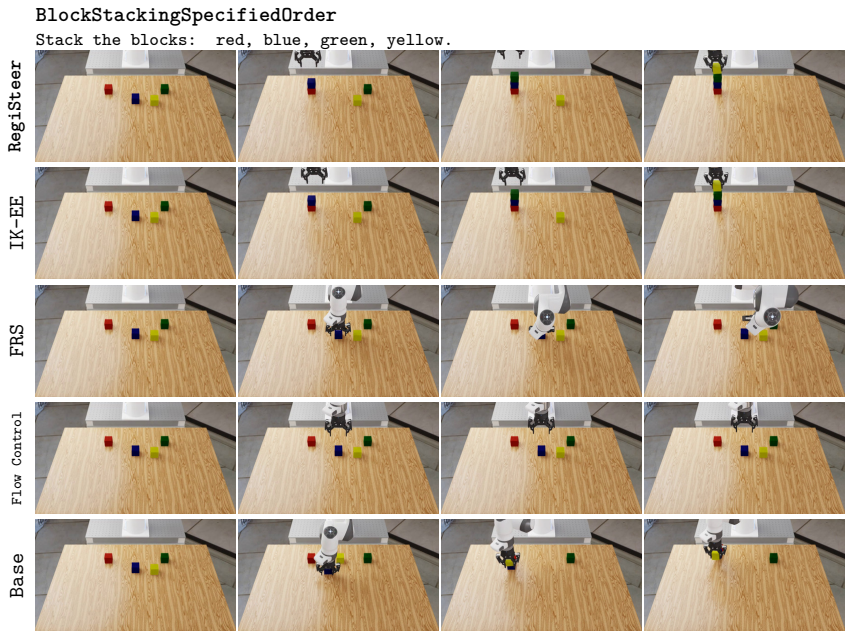


Figure 21: **`RegiSteer` and `IK-EE` complete the ordered stack.** `FRS` reaches the blue block without stacking it; `Flow Control` hovers above it, and the base policy leaves the stack unfinished.

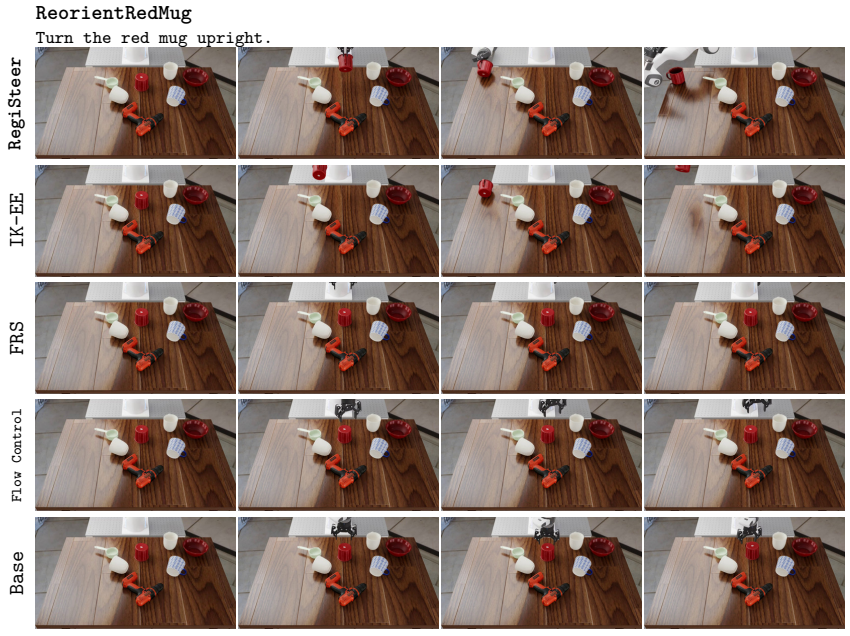


Figure 22: **`RegiSteer` turns the mug upright.** `IK-EE` and the base policy leave it inverted; `FRS` fails to grasp it, while `Flow Control` stalls above it.

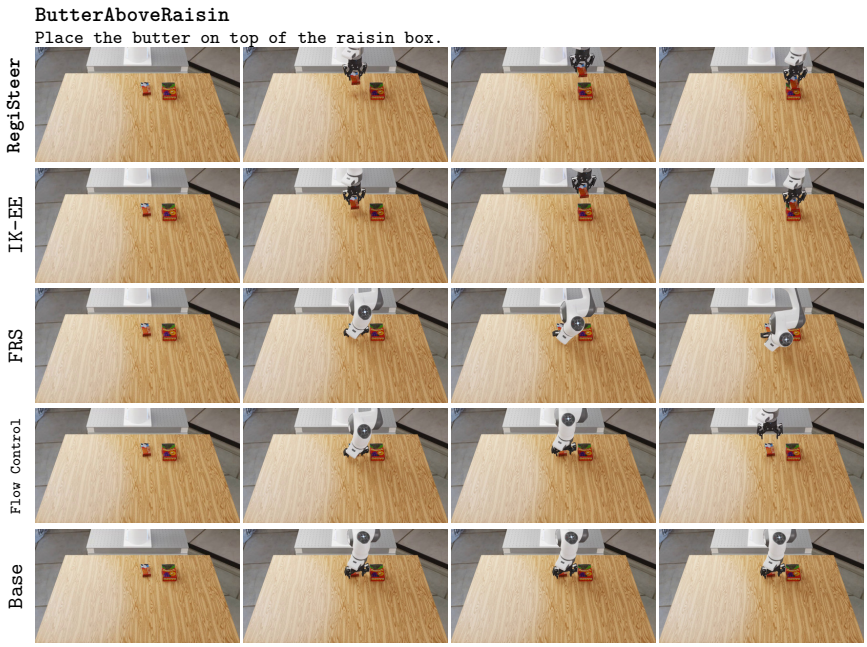


Figure 23: **`RegiSteer` and `IK-EE` place the butter on the raisin box.** `FRS`, `Flow Control`, and the base policy approach the butter but leave it off the box.



Figure 24: **`RegiSteer` transfers pumpkins from clutter to the bin.** `IK-EE` holds a pumpkin away from the bin; `FRS` and `Flow Control` stall near the objects, while the base policy picks other fruit.



Figure 25: **All four steering methods transfer fruit to the plate.** `RegiSteer`, `IK-EE`, `FRS`, and `Flow Control` make repeated transfers, while the base policy handles distractor objects.

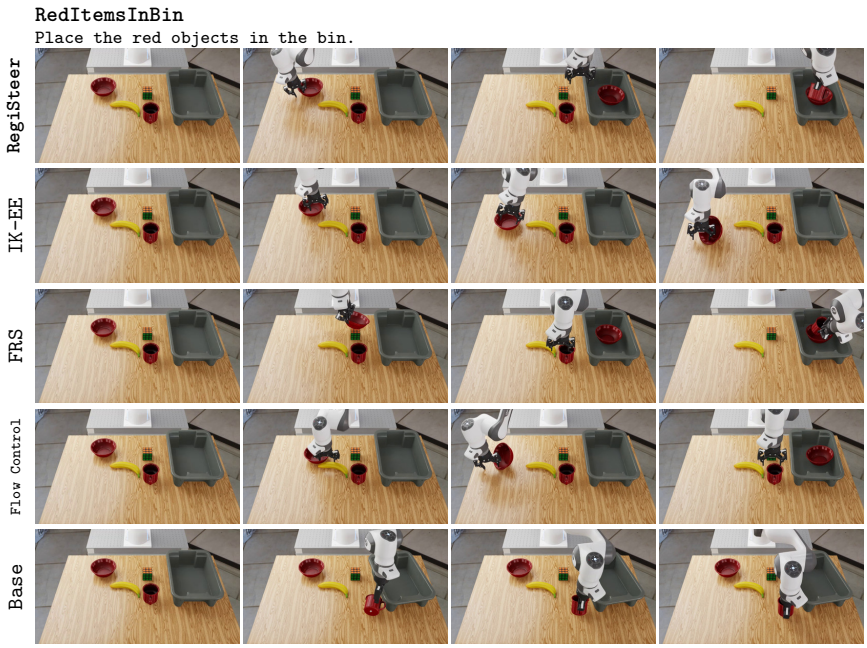


Figure 26: **`RegiSteer` and `FRS` transfer both red objects to the bin.** `IK-EE` leaves the bowl outside the bin; `Flow Control` and the base policy do not complete both placements.

Failure cases. Figures 27 and 28 show unsuccessful `RegiSteer` rollouts: partial sorting followed by lifting the bin, and repeated bottle grasps without placement.

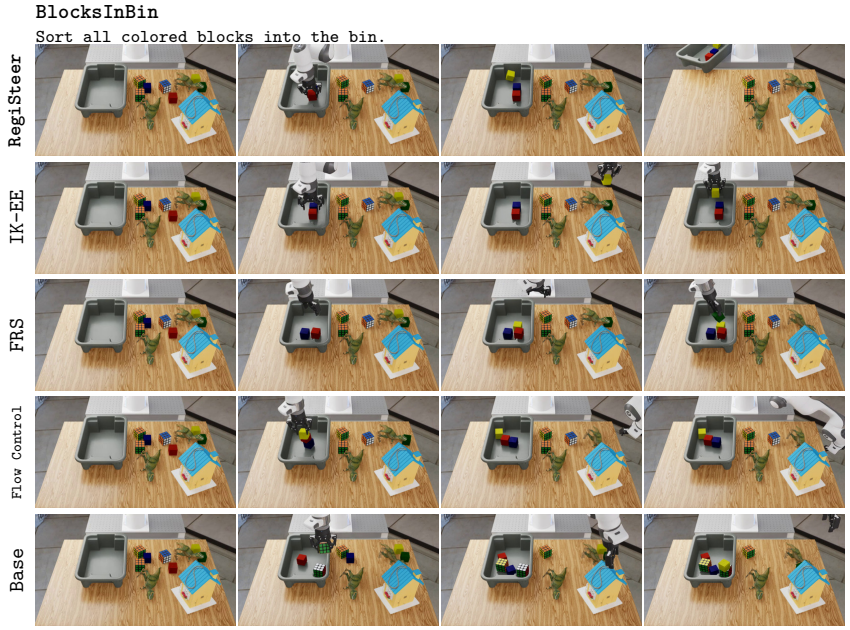


Figure 27: **`RegiSteer` leaves the green block unsorted and lifts the bin.** `FRS` completes sorting; `IK-EE` holds the yellow block, `Flow Control` leaves the green block out, and the base policy also collects distractors.



Figure 28: **`RegiSteer` lifts the bottle but leaves it outside the crate.** `IK-EE` holds it aloft; `FRS` stalls near it, `Flow Control` hovers above it, and the base policy drops it beside the crate.

A.8 ADDITIONAL LIBERO QUALITATIVE RESULTS

Figures 29 to 32 compare `RegiSteer`, IK-EE, and the base OpenVLA policy on one task from each LIBERO suite under LIBERO-PRO `swap`. Each row shows four chronological agent-view frames from the same relocated-object layout, selected independently per rollout. FRS and Flow Control require diffusion/flow action heads and do not apply to OpenVLA's autoregressive decoder.

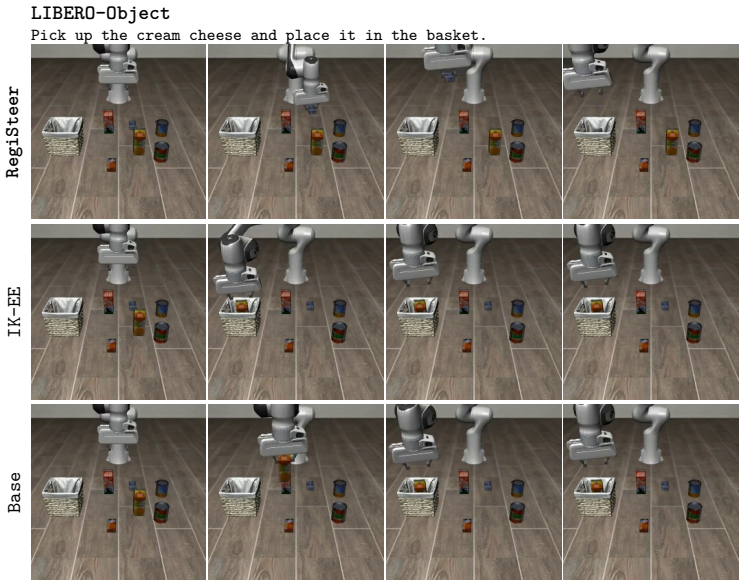


Figure 29: **`RegiSteer` reaches the relocated cream cheese.** `IK-EE` and the base policy instead pick a different object and place it in the basket.

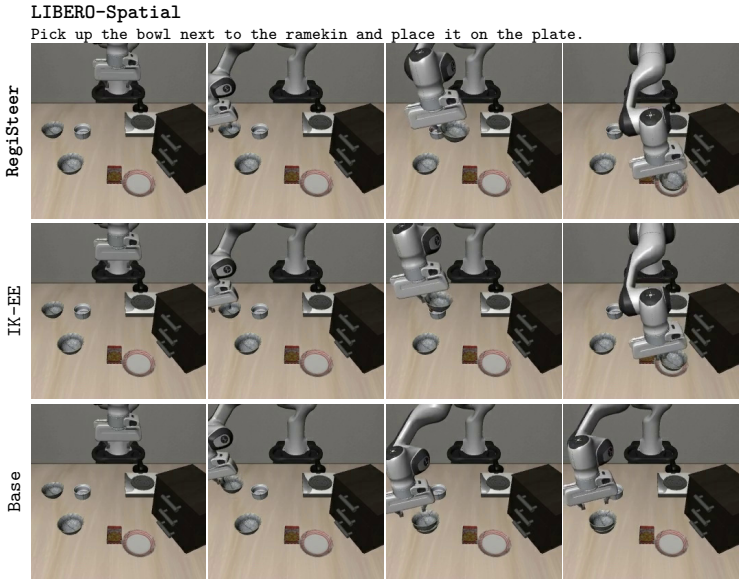


Figure 30: **`RegiSteer` and `IK-EE` select the requested bowl.** They move the bowl beside the ramekin to the plate; the base policy moves a different bowl.

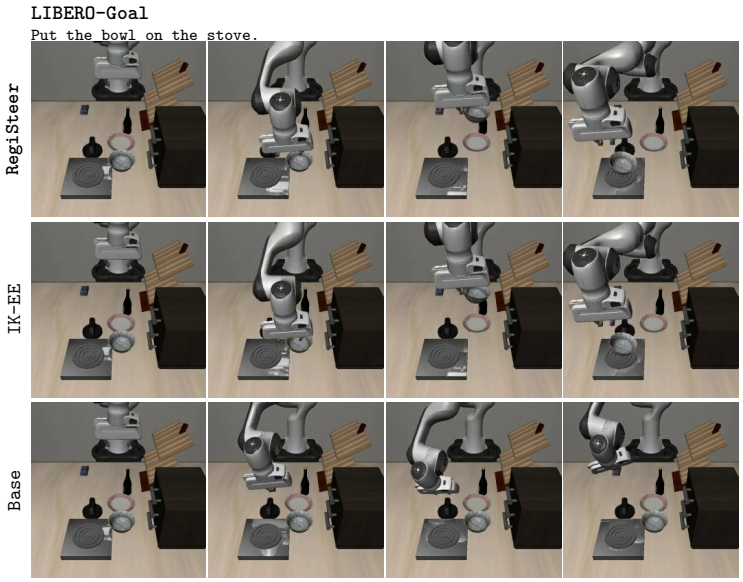


Figure 31: **`RegiSteer` and `IK-EE` reach the relocated stove.** Both place the bowl on the stove, while the base policy leaves it off the stove.



Figure 32: **`RegiSteer` completes both placements in the basket.** `IK-EE` stops after the first object, while the base policy reaches toward the wrong locations.

A.9 DIRECT VISUAL CONTROL WITH ASTRA

We expand the comparison in Fig. 7 between $\pi_{0.5}$+RegiSteer and GPT-6-Astra with high reasoning (OpenAI, 2026). Astra serves as an end-to-end robot controller, directly predicting motions from visual observations, whereas RegiSteer steers a pretrained VLA while retaining its learned manipulation behavior.

Experimental procedure. For the seven tasks below, Astra receives the task instruction, wrist and shoulder RGB images, and command history, and outputs a short visual rationale followed by seven joint targets and binary gripper commands. A simulator bridge interpolates these targets at 15 control steps per second and returns new observations after each action sequence. Astra receives no scene geometry and uses no VLA, inverse-kinematics solver, or external perception or planning module. We retain the DROID embodiment and native RoboLab task budgets. The simulation clock is paused during inference, so reasoning time does not consume the control budget.

Quantitative results. Table 11 reports object counts from six RegiSteer and three Astra trials per task. OBJ denotes the number of credited target placements divided by the total number of target opportunities, while SR denotes the fraction of trials in which all targets are credited. Overall OBJ pools object counts across tasks rather than averaging per-task percentages. Counts follow the task scoring protocol and do not necessarily indicate that every object was released and fully settled, as illustrated in Fig. 38.

Table 11: **RegiSteer achieves more credited object placements overall.** Entries show the object count for each trial, with Targets denoting the required number per trial. OBJ and SR are percentages; the higher value for each task is shaded, with ties shaded for both methods.

		RegiSteer			Astra		
Task	Targets	Counts	OBJ(%)	SR(%)	Counts	OBJ(%)	SR(%)
ClutterPlastic	3	2,3,2,3,2,3	83.3	50.0	2,2,3	77.8	33.3
RedItemsInBin	2	2,1,1,2,2,0	66.7	50.0	1,2,2	83.3	66.7
CondimentsInBin	4	4,3,2,3,2,2	66.7	16.7	3,1,1	41.7	0.0
ToolOrganizationBoth	2	1,2,0,1,0,1	41.7	16.7	0,0,0	0.0	0.0
CannedFoodInBin	1	1,1,1,1,1,1	100.0	100.0	1,1,0	66.7	66.7
ButterAboveRaisin	1	1,1,1,1,1,1	100.0	100.0	1,1,1	100.0	100.0
LargerObjectRaisinBoxInBin	1	1,1,1,1,0,1	83.3	83.3	0,0,0	0.0	0.0
Overall			72.6	59.5		52.4	38.1

RegiSteer receives credit for 61 of 84 target opportunities, compared with 22 of 42 for Astra, and completes 25 of 42 trials, compared with 8 of 21. This corresponds to overall gains of 20.2 percentage points in OBJ and 21.4 points in SR. The largest task-level differences occur in tool organization and larger-box placement. Astra performs better on red-item sorting, while both methods receive full credit on butter placement. Thus, RegiSteer's overall advantage is consistent across the aggregate metrics but does not extend to every individual task.

Qualitative rollouts and visual rationales. Figures 33 to 38 show Astra rollouts with four chronological shoulder-view frames each, cropped consistently to the workspace. The complete saved rationale for each selected observation appears below its caption. Rationales describe Astra's observation or intended next action, rather than verified outcomes. Frames precede the corresponding actions; captions summarize the full rollout outcomes.

`ClutterPlastic`

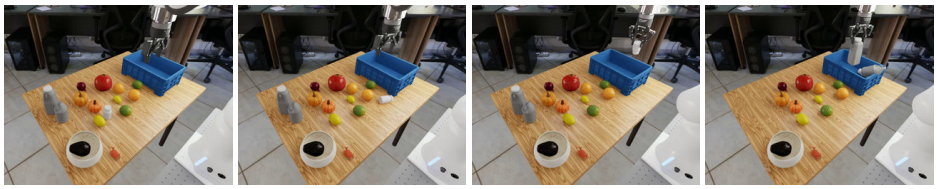


Figure 33: **Astra recovers a missed placement and transfers all three bottles.** The small bottle first lands outside the bin, then is recovered. The episode ends with the final jug released over the bin, before its settling is observed.

Frame 1. "Move open hand toward central plastic bottle while staying above clutter"
Frame 2. "Bottle settled on tabletop outside bin; approach for recovery"
Frame 3. "Release recovered bottle over bin interior"
Frame 4. "Release final milk jug into bin and allow settling"

`RedItemsInBin`

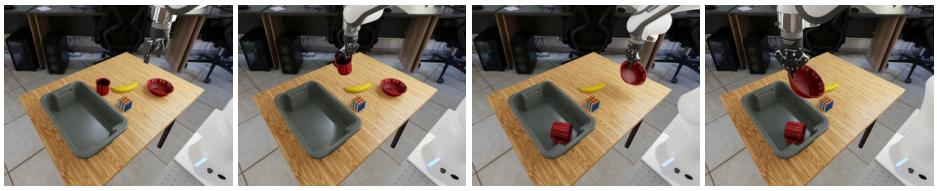


Figure 34: **Astra transfers the red cup and bowl to the bin.** After an initial missed cup grasp, it lifts the cup by its rim, then grasps and transfers the bowl. The task receives credit for both objects.

Frame 1. "The open gripper is elevated above the red bowl region. Lower and extend slightly while maintaining a downward wrist."
Frame 2. "Both cameras show the cup lifted with the gripper. Swing toward the grey bin while holding the rim."
Frame 3. "Bowl is visibly lifted by its rim. Raise slightly and transfer it over the bin while retaining the grasp."
Frame 4. "Bowl hangs near the bin edge. Move farther over its interior, lower gently, release, and retreat."

`CannedFoodInBin`



Figure 35: **Astra grasps the can and carries it into the bin.** Visual feedback guides alignment, lifting, and transfer. Native success occurs during the release command.

Frame 1. "The small round metal can lies between mustard and bowl near bin. Lower the open gripper toward that can while shifting modestly across the table."
Frame 2. "Can is centered between jaws. Lower slightly to its sidewall, close, then lift for visual confirmation."
Frame 3. "Held can is above the bin opening but near an edge in wrist view. Move farther over the interior and lower before releasing."
Frame 4. "Can is above the bin floor with clearance from walls in shoulder view. Open jaws and withdraw to detach."

Failure cases. Figures 36 to 38 show Astra's placement failures: dropped hammers, a raisin box caught in the gripper, and butter held above the target without release. The butter rollout receives native success credit despite the incomplete placement.

`ToolOrganizationBoth`

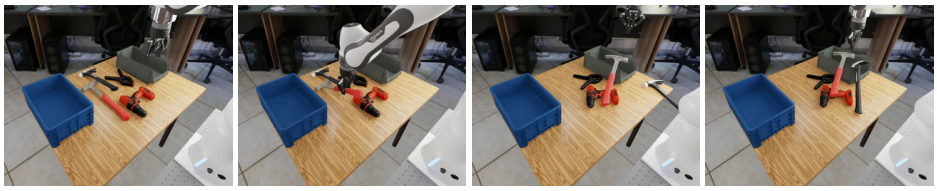


Figure 36: **Repeated slips prevent Astra from placing either hammer.** The red hammer slips during transfer, followed by the black hammer. Both finish leaning outside the right bin, and other tools are displaced during recovery attempts.

Frame 1. "Move the open gripper out above the hammer handles, keeping clear of the drill and pliers."
Frame 2. "The red hammer is visibly held; lift clear, then move above the gray bin on the right."
Frame 3. "The black hammer slipped onto the right tabletop edge, with its head still supported. Approach the upright red handle for recovery while monitoring the black hammer."
Frame 4. "The black hammer slipped safely back onto the tabletop against the bin. Use the remaining time for a short handle regrasp and inward lift."

`LargerObjectRaisinBoxInBin`



Figure 37: **Astra lifts the larger raisin box but does not complete placement.** After several grasp attempts, it carries the box over the bin. The box remains caught in the gripper, and the episode times out without placement credit.

Frame 1. "Fresh scene shows larger raisin box in foreground beside butter. Approach foreground with a folded elbow, maintaining clearance from the bin."
Frame 2. "The raisin is visibly lifted in the closed fingers. Extend the arm toward the grey bin while maintaining height and closure."
Frame 3. "Both views show the held raisin above the grey bin interior. Open the gripper to place it, then withdraw slightly while allowing it to settle."
Frame 4. "The box remains wedged between open-command fingers after release. Lower it into the bin and vary wrist alignment slightly to let the receptacle support it and free the jaws within remaining budget."

`ButterAboveRaisin`



Figure 38: **Astra does not let go of the butter.** It lifts the butter and lowers it above the raisin box. Native success terminates the episode with the gripper closed, so these frames do not demonstrate a released, stable stack.

Frame 1. "Move open fingers toward butter from the initial elevated pose"
Frame 2. "Butter lies between the fingers; close then lift to verify the grasp"
Frame 3. "Carry the grasped butter laterally over the raisin box"
Frame 4. "Center butter over raisin and lower it toward the top surface"

A.10 AI DISCLOSURE INFORMATION

We evaluated GPT-6-Astra, a generative AI model, as a robot-control comparison (Section A.9). We also used generative AI tools to write code for automating ablation experiments. Additionally, we used these tools to refine the writing and layout of the submission. We have reviewed all AI-assisted work and double-checked the experimental results produced using the automation scripts. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.