Object-Centric Residual RL for Zero-Shot Sim-to-Real VLA Enhancement

Kinam Kim[1,2,†], **Namiko Saito**[2], **Heecheol Kim**[2], **Katsushi Ikeuchi**[2,3], **Jaegul Choo**[1], **Yasuyuki Matsushita**[2]
[1] KAIST, South Korea, [2] Microsoft Research Asia - Tokyo, Japan
[3] The University of Tokyo, Japan

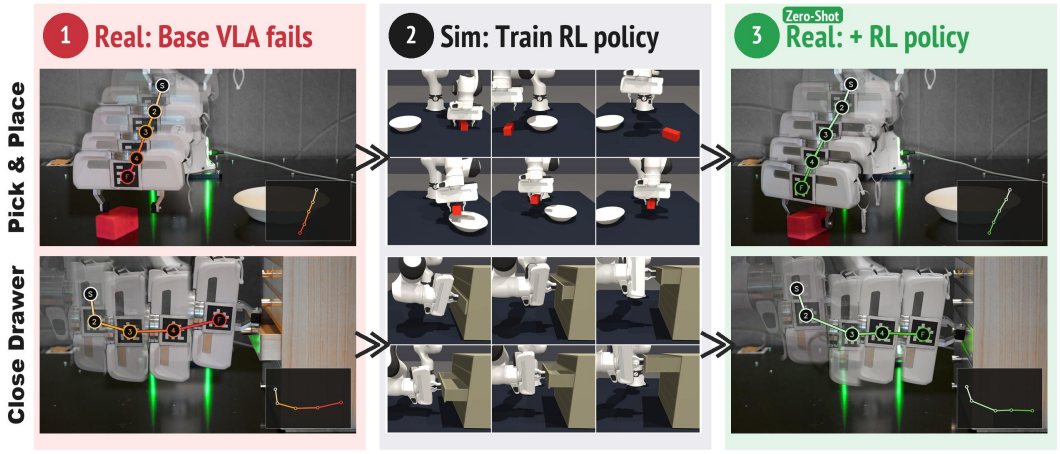


Figure 1: **Object-centric residual RL for zero-shot sim-to-real VLA enhancement.** The base VLA fails on the real robot (**left**). A residual policy trained purely in simulation **(middle)** is added zero-shot to recover task success on the same real-robot setup (**right**).

Abstract: Vision-Language-Action (VLA) models can generalize across diverse manipulation tasks, but their imitation-learning-based policies remain brittle in precise physical interactions due to compounding execution errors; *Can a reinforcement learning policy trained purely in simulation improve the robustness of real-world VLAs zero-shot?* Residual RL, which learns a corrective policy on top of a frozen VLA, offers a natural framework, but existing approaches face a fundamental sim-to-real dilemma: privileged-state methods require lossy distillation for deployment; image-based methods suffer from the visual domain gap; and real-world RL is costly and unsafe. We propose an object-centric residual RL framework that refines VLA actions using object poses, enabling a compact observation space that transfers consistently between simulation and reality. To align the two domains, we additionally replay the same teleoperation demonstrations in simulation to train a sim counterpart of the real-world VLA. The residual RL policy is trained only in simulation with pose noise injection and dropout, and transfers zero-shot to the real robot. Across five manipulation tasks on a real Franka Research 3 (FR3) robot, our method improves the success rate from 42% to $\mathbf{76}\%$ zero-shot, and the improved rollouts can be further reused to retrain the base VLA for self-improvement without additional teleoperation. Project page: https://www.microsoft.com/en-us/research/articles/object-centric-residual-rl/

Keywords: Vision-Language-Action Models, Reinforcement Learning, Sim-to-Real Transfer, Robot Manipulation

[†]Work done during an internship at Microsoft Research Asia.

1 Introduction

Vision-Language-Action models (VLAs) enable broad manipulation capabilities by leveraging large-scale pretraining and robot demonstrations [1–8]. However, because VLAs are trained via imitation learning, small errors accumulate over time and lead to failures in unseen states [9, 10]. Reinforcement learning (RL) can improve recovery through online interaction, but directly applying RL to modern VLAs is difficult because many architectures rely on diffusion [11, 12] or flow matching [6, 13] for action generation, whose iterative denoising is not readily differentiable through standard policy gradients. Residual RL [14, 15] addresses this by learning a lightweight corrective policy on top of a frozen VLA, but sim-to-real transfer of the residual remains a major challenge.

Three approaches have been pursued regarding residual RL, each with a distinct failure mode. **Distillation-based residual RL** [16] trains on privileged simulator state and requires teacher-student *distillation* into an image-based student for deployment, incurring performance loss. **Image-based sim residual RL** avoids distillation by operating directly on images, but suffers from a large visual sim-to-real domain gap that prevents zero-shot transfer of the sim-trained residual. **Real-world residual RL** [17, 18] eliminates the need for sim-to-real transfer by training directly on the real robot, but is costly and raises safety concerns [19]. None of these paradigms achieves zero-shot transfer of a sim-trained residual policy to a real robot.

In this work, we observe that prior sim-trained residual policies fail to transfer because their observation spaces are inherently domain-dependent: privileged-state methods [16] rely on quantities unavailable on the real robot and must distill into an image policy, while image-based methods face a large visual sim-to-real gap. We take a different approach: rather than explicitly bridging them, we substantially reduce the discrepancies seen by the residual policy by building the residual on observations that are consistently recoverable in both domains—6-DoF object poses, proprioceptive state, and the base VLA action. Object pose can be reliably obtained via off-the-shelf estimators [20, 21], and because the residual operates on this low-dimensional state rather than images, it transfers **zero-shot** without distillation or real-world RL. To align action distributions across domains, we replay the same teleoperation trajectories in simulation to train a sim VLA alongside the real one.

Beyond zero-shot deployment, our framework also enables automatic VLA self-improvement. Successful real-robot rollouts collected by deploying the residual-corrected policy can be aggregated across tasks to retrain a single multi-task VLA, producing higher-quality training data without any additional teleoperation.

Our contributions are as follows:

- We propose a **zero-shot sim-to-real residual RL framework** with two design choices: paired sim and real VLAs aligned via teleoperation replay, and a **domain-invariant residual interface** that sidesteps the sim-to-real gaps without distillation or real-world RL.
- We show that real-robot rollouts from the residual-corrected policy can be aggregated across tasks to retrain a single **multi-task VLA**, enabling **automatic self-improvement** without additional teleoperation.
- We demonstrate consistent improvement across **five manipulation tasks** in both simulation and **real-world zero-shot deployment**, with an average improvement from 42% to $\mathbf{76}\%$ success rate on a real FR3 robot.

2 Related Work

Residual Reinforcement Learning. Residual RL [14, 15] trains a corrective policy on top of a frozen base, combining the generalization of the base with the precision of RL. ResiP [16] trains the residual policy on privileged simulator state, achieving strong sim performance but requiring a separate teacher-student distillation step for real-world deployment, which incurs a non-trivial performance loss during the transfer. RialTo [22] avoids real-world RL by constructing a digital twin via 3D scanning of the entire workspace and training point-cloud-based policies through an inverse

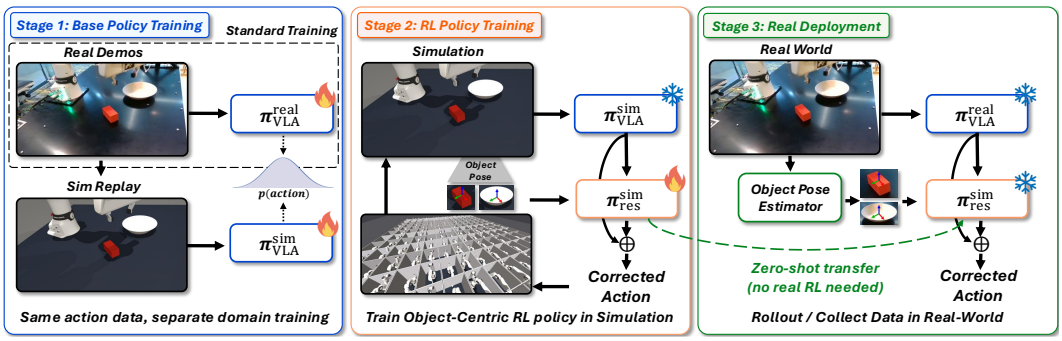


Figure 2: Overview of the object-centric residual RL pipeline.

distillation pipeline, but the end-to-end process (3D reconstruction, scene registration, point-cloud policy training, and inverse distillation) incurs substantial per-task time and engineering overhead, limiting scalability to new tasks. ResFiT [17] operates directly on images and proprioception, but must train on the real robot, requiring safe exploration infrastructure, episode resets, and real-world reward detection. PLD [18] extends this to VLAs with a three-stage probe-learn-distill pipeline, also requiring real-world RL and an additional reward classifier. RPD [23] distills VLA knowledge into an RL student entirely in simulation, but has not demonstrated real-world deployment. In contrast, our method operates on object pose, a compact representation accurately recoverable in reality, enabling zero-shot sim-to-real transfer without distillation, real-world RL, or complex reconstruction pipelines.

Sim-to-Real Transfer. Bridging the gap between simulation and reality has been a longstanding challenge [24]. Domain randomization [25–27] varies visual and physical parameters during training to promote robustness, and has enabled impressive sim-to-real results including dexterous in-hand manipulation [28]. Recent work automates this process with language models [29], while others adapt randomization distributions using real-world data [30, 31] or learn residual corrections from online human feedback [32]. When the policy uses privileged simulator state, a common recipe is teacher-student distillation [16, 33], where a state-based teacher is first trained in simulation and then distilled into a vision-based student for real-world deployment. We take an orthogonal approach: instead of aligning visual or dynamics distributions across domains, we design the residual's observation space to remain consistent between simulation and reality, enabling zero-shot transfer without domain adaptation or distillation.

3 Method

Given a VLA trained on real-robot demonstrations, our framework enhances it with a sim-trained residual policy through three stages (Fig. 2): (1) building a simulation counterpart of the VLA via teleoperation replay (Section 3.1), (2) training an object-centric residual policy in simulation (Section 3.2), and (3) zero-shot deployment on the real robot (Section 3.3). The combined deployment action is

$$a_t = a_t^{\text{base}} \oplus \pi_{\text{res}}^{\text{sim}}(s_t), \tag{1}$$

where a_t^{base} is the current base action drawn from the real VLA's chunked rollout (Stage 1), and $\pi_{\text{res}}^{\text{sim}}$ is the sim-trained residual queried at every timestep t on object-centric observation s_t (Stage 2). The VLA takes an RGB observation, proprioceptive state, and language instruction; the VLA is kept frozen during residual training, and both are frozen at deployment. $\oplus$ denotes per-component action composition: addition for position and gripper, quaternion multiplication for rotation. The residual is trained to consistently improve task success over π_{VLA} alone, and deploys zero-shot on the real robot without any real-world RL or distillation. Beyond deployment, the residual-corrected policy can also be used to retrain the base VLA itself, enabling multi-task generalization of per-task residual-RL behaviors (Section 3.4).

3.1 Stage 1: Paired Sim/Real VLA via Teleoperation Replay

Standard task-specific VLA fine-tuning collects teleoperation demonstrations on the real robot and trains a VLA on real (RGB, action) pairs. We extend this standard recipe by additionally replaying the same teleoperation actions in a simulation environment [34], rendering a parallel sim dataset that trains a sim VLA $\pi_{\text{VLA}}^{\text{sim}}$ alongside the standard real VLA $\pi_{\text{VLA}}^{\text{real}}$. Because both VLAs are supervised by identical teleoperation actions, they learn aligned action distributions despite seeing different visual domains, letting the residual train alongside $\pi_{\text{VLA}}^{\text{sim}}$ in simulation and transfer to $\pi_{\text{VLA}}^{\text{real}}$ zero-shot at deployment. In short, $\pi_{\text{VLA}}^{\text{sim}}$ provides a_{base} for residual RL training in simulation, while $\pi_{\text{VLA}}^{\text{real}}$ replaces it at deployment. Since the residual policy does not observe images, the simulation need not be visually realistic, significantly reducing the engineering effort required to construct the simulation environment.

3.2 Stage 2: Object-Centric Residual RL

The key challenge is choosing the observation s for $\pi_{\text{res}}^{\text{sim}}$. If s contains privileged simulator state (e.g., contact forces), transfer fails. If s contains images, the visual domain gap prevents zero-shot transfer. We resolve this by constructing s from quantities that are both informative and accurately recoverable in reality.

Observation space. In our work, the residual policy observation is defined as:

$$s_t = [\, s_t^{\text{obj}},\ s_t^{\text{prop}},\ a_t^{\text{base}} \,], \tag{2}$$

where s_t^{obj} is the 6-DoF pose (position and orientation) of the task-relevant objects (manually specified in the current setup), s_t^{prop} is the proprioceptive state, and a_t^{base} is the current base action drawn from the VLA's action chunk. For tasks involving multiple objects (e.g., Stack Cube requires tracking both the grasped cube and the target cube), the poses of all task-relevant objects are concatenated into s_{obj}. Our residual formulation enables continued improvement via RL while the VLA remains frozen, decoupling the pose-based correction from the base policy's training. We note that our approach is orthogonal to recent work on 3D-conditioned VLAs [35–37]: if such a model serves as the base policy, our residual module can still be applied on top to provide additional RL-based corrections. Note that the framework is also agnostic to the specific proprioceptive representation (e.g., end-effector pose vs. joint angles) and to the action parameterization of the base VLA.

Zero-shot transfer condition. We formalize the condition under which a sim-trained residual policy can transfer zero-shot. Let s^{sim} and s^{real} denote the observation vectors in simulation and reality, respectively. Zero-shot transfer becomes feasible when the residual policy is robust to the deployment noise $\mathcal{P}_\eta$:

$$s_t^{\text{real}} = s_t^{\text{sim}} + \eta_t, \quad \eta_t \sim \mathcal{P}_\eta. \tag{3}$$

The magnitude of $\mathcal{P}_\eta$ depends on the choice of observation s_t. If s_t includes privileged simulator state (e.g., contact forces), η_t is large due to systematic physics modeling errors. If s_t includes images, η_t is large due to rendering discrepancies in lighting, textures, and backgrounds. Our observation space minimizes $\mathcal{P}_\eta$ by construction:

- Proprioception (end-effector pose and gripper state): domain-invariant, contributing $\eta \approx 0$.
- Base VLA action: $\pi_{\text{VLA}}^{\text{sim}}$ and $\pi_{\text{VLA}}^{\text{real}}$ are trained on the same teleoperated trajectories, yielding closely aligned outputs ($\eta \approx 0$).
- Object pose: estimated via FoundationPose [20] with SAM2 [21]; the only component with non-negligible η_t.

We address the remaining pose estimation noise $\mathcal{P}_\eta$ by augmenting the pose input with structured noise during training, as detailed below.

Robust object pose training. Real-world pose estimators introduce noise and occasional failures. To ensure that the residual policy is robust to these errors, we apply two forms of augmentation

during training. We decompose the 6-DoF object pose s_{obj} into position p_{obj} and orientation q_{obj} components. First, at each timestep we perturb each component with random noise:

$$\hat{p}_{\text{obj}} = p_{\text{obj}} + \epsilon_p, \quad \hat{q}_{\text{obj}} = q_{\text{obj}} \otimes \epsilon_q, \tag{4}$$

where $\otimes$ denotes quaternion multiplication, each component of ϵ_p is independently sampled from $\mathcal{U}(-\tilde{\sigma}_p, \tilde{\sigma}_p)$ with $\tilde{\sigma}_p \sim \mathcal{U}(0, \sigma_p^{\max})$ resampled per timestep, and ϵ_q is a small random rotation with magnitude similarly drawn via $\tilde{\sigma}_q \sim \mathcal{U}(0, \sigma_q^{\max})$. This hierarchical uniform sampling exposes the policy to a continuous range of small pose perturbations, mirroring the varying accuracy of real-world pose estimators. We denote the combined noise-augmented pose as $\hat{x}_{\text{obj}} = (\hat{p}_{\text{obj}}, \hat{q}_{\text{obj}})$. Second, with probability ρ_{drop} we zero out the entire object pose vector:

$$\tilde{x}_{\text{obj}} = \begin{cases} \hat{x}_{\text{obj}} & \text{with probability } 1 - \rho_{\text{drop}} \\ \mathbf{0} & \text{with probability } \rho_{\text{drop}}. \end{cases} \tag{5}$$

This forces the policy to learn a fallback strategy using only proprioception and the base action, ensuring graceful degradation when the pose estimator fails.

Reinforcement learning. We train the residual policy using TD3 [38] with clipped exploration noise for stable off-policy optimization. Exploration adds clipped Gaussian noise on top of the combined action, while pose-noise scales $(\sigma_p^{\max}, \sigma_q^{\max})$ and dropout probability ρ_{drop} control the augmentation magnitudes. $\pi_{\text{VLA}}^{\text{sim}}$ is queried every H steps to produce an H-length action chunk

$$A_k = \pi_{\text{VLA}}^{\text{sim}}(o_{kH}^{\text{img}}, s_{kH}^{\text{prop}}, l), \quad k = \lfloor t/H \rfloor, \tag{6}$$

where o_{kH}^{img} and s_{kH}^{prop} are the RGB observation and proprioceptive state at chunk timestep kH, and l is the language instruction. From the resulting chunk A_k, we read the current base action $a_t^{\text{base}} = A_k[t \bmod H]$. The residual $\pi_{\text{res}}^{\text{sim}}$ is queried at every timestep t with observation s_t from Eq. (2), yielding the combined action:

$$a_t = a_t^{\text{base}} \oplus \pi_{\text{res}}^{\text{sim}}(s_t). \tag{7}$$

The residual policy is trained with dense rewards (see Appendix A.1 for details).

3.3 Stage 3: Zero-Shot Deployment

At deployment time, both $\pi_{\text{VLA}}^{\text{real}}$ and $\pi_{\text{res}}^{\text{sim}}$ are frozen; the system requires no further training or adaptation. The base action a_t^{base} is read from $\pi_{\text{VLA}}^{\text{real}}$'s action chunk analogously to Eq. (6), and the combined action follows Eq. (7). At each timestep, we additionally consult the pose estimator's tracking confidence c_t: when c_t falls below a threshold τ_c, the pose input is zeroed out to trigger the dropout fallback learned during training (Eq. (5)). The residual module adds minimal computational overhead: the actor forward pass takes less than 1 ms, and FoundationPose runs in tracking mode after initial registration via SAM2 [21] ($\sim$18 ms per frame); we run it asynchronously so that pose estimation does not bottleneck the control loop. The confidence-gated dropout bridges training and deployment: the random dropout at training time prepares the policy for the *systematic* dropout that occurs at deployment when poses are lost.

3.4 VLA Self-Improvement

Beyond deployment-time enhancement, successful rollouts from the residual-corrected policy can be merged with the original demonstrations to retrain the base VLA via supervised fine-tuning. This self-improvement loop requires no additional data collection with teleoperation; rollout data from multiple task-specific residuals can be aggregated to retrain a single multi-task VLA, preserving generalist ability.

4 Experimental Setup

We evaluate on five tabletop manipulation tasks, instantiated both in MuJoCo [39] simulation (for residual training) and on a real FR3 robot (for zero-shot deployment):

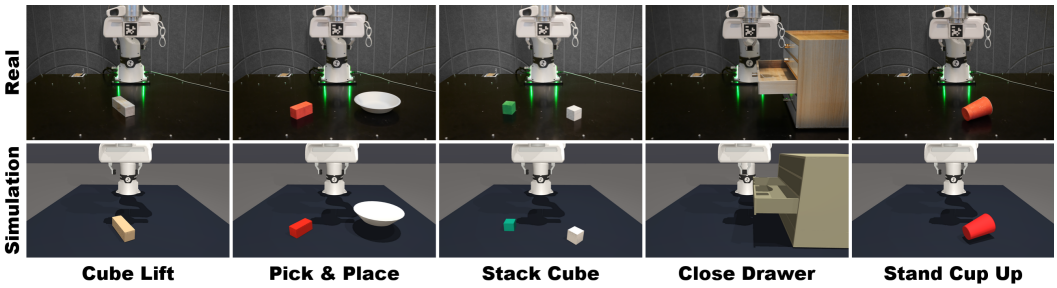
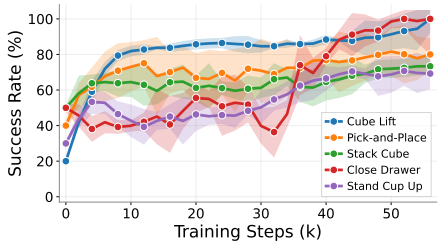


Figure 3: Real (top) and simulated (bottom) environments for all five evaluation tasks.

Table 1: Success rates in simulation and real-robot. Simulation results are reported as mean ± standard deviation over 3 seeds.

	Simulation		Real Robot	
Task	Base	+Res.	Base	+Res.
Cube Lift	$4.3/20 \pm 0.6$	$\mathbf{20.0/20} \pm 0.0$	$7/20$	$\mathbf{17/20}$
Pick-and-Place	$7.0/20 \pm 2.6$	$\mathbf{17.0/20} \pm 2.0$	$9/20$	$\mathbf{16/20}$
Stack Cube	$10.0/20 \pm 1.0$	$\mathbf{14.7/20} \pm 0.6$	$7/20$	$\mathbf{15/20}$
Close Drawer	$11.3/20 \pm 3.2$	$\mathbf{19.7/20} \pm 0.6$	$14/20$	$\mathbf{20/20}$
Stand Cup Up	$5.3/20 \pm 1.2$	$\mathbf{14.7/20} \pm 1.2$	$5/20$	$\mathbf{8/20}$
Average	$7.6/20 \pm 1.7$	$\mathbf{17.2/20} \pm 0.9$	$8.4/20$	$\mathbf{15.2/20}$

Figure 4: Success rates across 3-seed training in simulation. Shaded regions denote standard deviation.



1. **Cube Lift (Lift)**: Grasp a cube from the table and lift it 3 cm.
2. **Pick-and-Place (PnP)**: Pick up a cube and place it into a bowl.
3. **Stack Cube (Stack)**: Pick up a white cube and stack it on top of a green cube.
4. **Close Drawer (Close)**: Push an open cabinet drawer closed.
5. **Stand Cup Up (Stand)**: Grasp a cup lying on its side and stand it upright.

The simulation is built by measuring real object dimensions; visual realism is not required (Fig. 3). Further details on the simulation environment are provided in Appendix A.2. We use GR00T-N1.5 [8], an open-source VLA, as the base policy fine-tuned on 30 teleoperation demonstrations per task for each domain (sim and real) [40]. The residual policy is a lightweight 2-layer MLP trained with TD3 [38]; a single forward pass takes $\sim$0.06 ms on GPU, less than 0.05% of the VLA's $\sim$140 ms inference time.

5 Results

We design our experiments to answer the following questions:

1. Does the residual improve the base VLA zero-shot on a real robot? (Section 5.1)
2. What observation and robustness designs enable sim-to-real transfer? (Section 5.2)
3. When and how does the residual intervene? (Section 5.3)
4. Can residual-corrected rollouts bootstrap VLA self-improvement? (Section 5.4)

5.1 Main Results

We first examine whether a residual policy trained entirely in simulation can improve the base VLA on a real robot without any adaptation. Table 1 reports success rates across all five tasks. The residual policy improves all five tasks in simulation, with the largest gains where the base VLA struggles most. On the real robot, the sim-trained residual transfers zero-shot to all five tasks, raising the average success rate from 42% to 76% without any real-world RL or fine-tuning.

Table 2: Ablation studies on real-robot performance (successes / 20 trials). **(a)** Robustness training: pose dropout and noise injection both contribute; combined training yields the strongest sim-to-real transfer. **(b)** Observation space: object-centric poses transfer best by avoiding the visual domain gap.

	Lift	PnP	Stack	Close	Stand
Ours	**17/20**	**16/20**	**15/20**	**20/20**	**8/20**
(a) Robustness Training					
w/o noise injection	16/20	14/20	11/20	**20/20**	**8/20**
w/o pose dropout	13/20	12/20	10/20	16/20	7/20
w/o both	12/20	10/20	9/20	16/20	5/20
(b) Observation Space					
Distillation-based	9/20	4/20	8/20	**20/20**	5/20
Image-based	8/20	10/20	9/20	14/20	6/20

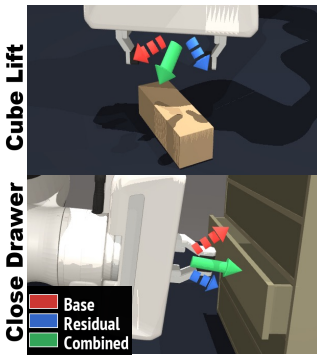


Figure 5: The residual corrects the base action toward the goal when misaligned.

Generalization across VLA architectures. To demonstrate that our residual RL framework is not specific to a single base VLA, we evaluate with $\pi_{0.5}$ [7]. As shown in Fig. 6(a), the residual RL consistently improves performance on the real robot, suggesting that the proposed object-centric observation interface is compatible with different VLA backbones.

5.2 Ablations

Robustness training. Table 2(a) ablates the two robustness mechanisms from Section 3.2: pose dropout contributes most strongly (resilience to detection failures), noise injection helps tight-tolerance tasks, and combining both yields the strongest transfer.

Observation space. Table 2(b) and Fig. 6(b) compare three observation designs. The image-based baseline suffers from the visual sim-to-real gap, and the distillation baseline, which distills a privileged-state teacher into an image-based student, loses performance during distillation. In contrast, our object-centric residual transfers best, indicating that the visual domain gap is the dominant sim-to-real barrier, which our object-centric observation sidesteps by design.

5.3 Analysis

Having established that the residual improves performance, we now investigate when and how it intervenes. Fig. 5 visualizes per-step action vectors for two tasks: the residual steers the combined action toward the goal when the base is misaligned. Across all five real-robot tasks (Fig. 7(a)), the residual consistently points toward the goal when the base is misaligned and contributes less when aligned, confirming selective correction; this translates to 9–22% faster task completion (Fig. 7(b)).

5.4 VLA Self-Improvement

Beyond direct deployment, supervised fine-tuning (SFT) of the base VLA on residual-corrected rollouts raises real-robot success rate and reduces episode length compared to SFT on plain base rollouts (Fig. 6(c,d)). These results suggest that residual-corrected trajectories provide higher-quality supervision for retraining the base VLA, enabling a self-improvement loop without additional teleoperation.

6 Conclusion

We presented object-centric residual RL, a method for enhancing Vision-Language-Action models through sim-trained residual policies that transfer to the real robot zero-shot. The key insight is that an object-centric observation space, constructed from 6-DoF object poses, proprioception, and the

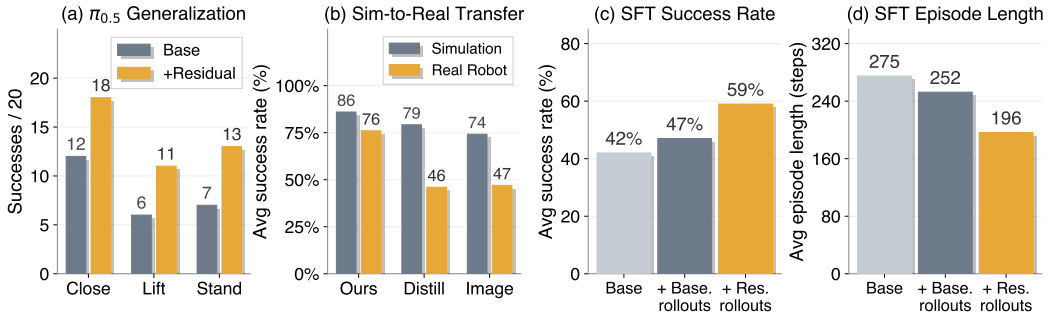


Figure 6: **(a)** Performance improvement on $\pi_{0.5}$ [7], demonstrating compatibility with different VLA backbones. **(b)** Sim-to-real transfer across observation spaces; the object-centric design transfers most effectively. **(c, d)** SFT on residual-corrected rollouts improves success rate and reduces episode length.

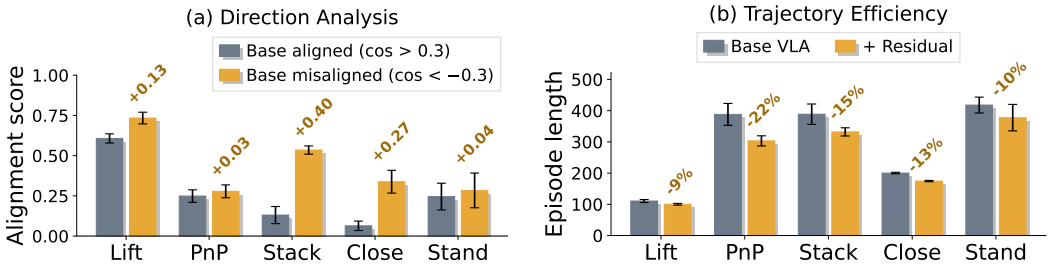


Figure 7: **(a)** Cosine similarity between the residual action and the goal direction, conditioned on base action alignment. The residual corrects more strongly when the base deviates. **(b)** Episode length comparison between base and residual-corrected policies (success episodes). The residual consistently reduces completion time by 9–22%. Error bars denote standard error of the mean across timesteps (a) and episodes (b).

base VLA action, is recoverable in both simulation and reality without visual rendering, enabling strong zero-shot sim-to-real transfer of the residual policy. Combined with robustness training via noise injection and pose dropout, our residual policies improve a GR00T-N1.5-based VLA across five tasks in both simulation and on a real FR3 robot, raising the real-robot average success rate from 42% to 76% without any real-world reinforcement learning or residual-policy fine-tuning. Beyond direct deployment, the residual-corrected policy generates improved real-robot rollouts that can be aggregated across tasks to retrain a single multi-task VLA, enabling a self-improvement loop that requires no additional teleoperation. We believe this paradigm, which combines the generalization of VLAs with the precise corrective capability of RL through a carefully chosen observation interface, offers a practical path toward scalable and autonomous robot improvement.

7 Limitations and Future Work

Our method relies on real-time 6-DoF pose tracking (FoundationPose + SAM2), which can fail under full occlusion or heavy clutter; memory-based pose estimation could mitigate this limitation. Task-relevant objects must also be specified manually; scaling to open-world settings would require automatic identification, e.g., from VLA attention maps. The pose-based observation bridges the visual domain gap but not the dynamics gap: contact friction and gripper compliance differences between sim and real may cause suboptimal corrections in contact-rich tasks. As a residual architecture, the policy can correct mild deviations but cannot recover from states far outside the base VLA's training distribution. Finally, tasks requiring sub-millimeter precision or involving very small objects may exceed the accuracy of current pose estimation; extending to such scenarios with higher-resolution sensing or tactile feedback is left to future work.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2023.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[3] Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[4] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[7] Physical Intelligence, K. Black, N. Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[8] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[9] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[10] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

[11] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

[12] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[13] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[14] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[15] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[16] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement – residual rl for precise assembly. In *Conference on Robot Learning (CoRL)*, 2024.

[17] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. ResFiT: Residual off-policy RL for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[18] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, et al. Self-improving vision-language-action models with data generation via residual RL. *arXiv preprint arXiv:2511.00091*, 2025.

[19] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, 2021.

[20] B. Wen, W. Yang, J. Kautz, and S. Birchfield. FoundationPose: Unified 6D pose estimation and tracking of novel objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[21] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, et al. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[22] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

[23] T. Jülg, W. Burgard, and F. Walter. Refined policy distillation: From VLA generalists to RL experts. *arXiv preprint arXiv:2503.05833*, 2025.

[24] W. Zhao, J. Peña Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. *arXiv preprint arXiv:2009.13303*, 2020.

[25] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[26] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2018.

[27] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, et al. DeXtreme: Transfer of agile in-hand manipulation from simulation to reality. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[28] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[29] Y. J. Ma, W. Liang, H.-J. Wang, S. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman. DrEureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024.

[30] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[31] F. Ramos, R. Possas, and D. Fox. BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *Robotics: Science and Systems (RSS)*, 2019.

[32] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. TRANSIC: Sim-to-real policy transfer by learning from online correction. In *Conference on Robot Learning (CoRL)*, 2024.

[33] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[34] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

[35] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3D Diffusion Policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems (RSS)*, 2024.

[36] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3D Diffuser Actor: Policy diffusion with 3D scene representations. In *Conference on Robot Learning (CoRL)*, 2024.

[37] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, et al. SpatialVLA: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[38] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

[39] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[40] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[41] Physical Intelligence et al. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Appendix for:
Object-Centric Residual RL for Zero-Shot Sim-to-Real VLA Enhancement

A Appendix

A.1 Reward Design

All tasks use dense, shaped rewards clipped to $[0, 1]$. Each reward is decomposed into staged sub-rewards that are applied progressively as the task advances. Table 3 summarizes the reward structure per task.

Table 3: Reward stages per task. Each stage provides a continuous signal based on distance, orientation, or contact metrics.

Task	Reward stages
Cube Lift	Reach $\rightarrow$ Grasp $\rightarrow$ Lift
Pick-and-Place	Reach $\rightarrow$ Grasp $\rightarrow$ Carry $\rightarrow$ Place
Stack Cube	Reach $\rightarrow$ Grasp $\rightarrow$ Align $\rightarrow$ Stack
Close Drawer	Push (closing progress) $\rightarrow$ Close (fully closed)
Stand Cup Up	Reach $\rightarrow$ Grasp $\rightarrow$ Upright

A.2 Simulation Environment Construction

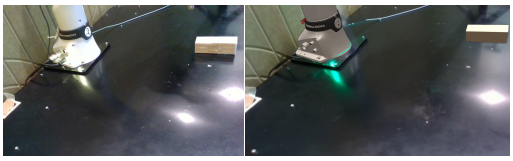
All tasks are built in MuJoCo [39]. Object dimensions and workspace layout are measured from the real setup; other scene parameters (table height, camera pose, etc.) do not require exact matching since the sim and real VLAs are trained separately and the residual policy does not observe images. Each object is modeled using a simple geometric primitive from the measured principal dimensions. For Cube Lift, Pick-and-Place, and Stack Cube, the cube is modeled as a simple box with measured side length; Pick-and-Place additionally includes a bowl, and Stack Cube places two cubes. For Close Drawer, the cabinet and drawer are modeled with a sliding joint whose range matches the real drawer travel. For Stand Cup Up, the cup is modeled as a truncated cone (approximated by stacked cylinder slices) placed on its side as the initial pose. Table 4 summarizes the object specifications used in simulation. During RL training, the initial position and orientation of task-relevant objects are randomized within the workspace to expose the residual policy to diverse configurations.

Table 4: Simulation object specifications. All dimensions are measured from the real objects; masses correspond to the values used in simulation. All cuboid blocks (Cube Lift, Pick-and-Place, Stack Cube) and the Close Drawer cabinet/drawers are modeled with a wood material; the Pick-and-Place bowl is a thin paper bowl; the Stand Cup Up cup is a rigid plastic-like shell.

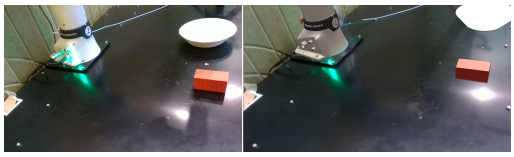
Task	Object	Geometry	Dimensions	Mass	Color
Cube Lift	Cuboid	box	$12 \times 4 \times 4$ cm	75 g	wood brown
Pick-and-Place	Cuboid (grasped) Bowl (target)	box cylinder	$8 \times 4 \times 4$ cm radius 9.5 cm, height 1 cm	50 g 10 g	red white
Stack Cube	Cube (grasped) Cube (target)	box box	$4 \times 4 \times 4$ cm $4 \times 4 \times 4$ cm	25 g 25 g	white green
Close Drawer	Cabinet Drawer ($\times 5$)	box composite slide joint	$27 \times 35 \times 28.2$ cm travel 13 cm	4.9 kg 0.87 kg	wood brown wood brown
Stand Cup Up	Cup	truncated cone	top $\varnothing$ 7.5 cm, bottom $\varnothing$ 5 cm, height 10 cm	150 g	red

A.3 Realistic Simulation Rendering

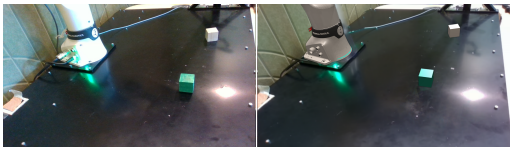
While our main method uses object-centric observations, we additionally set up a visually realistic MuJoCo rendering environment to enable image-based RL training and policy distillation as auxiliary baselines in simulation. Specifically, we use high-quality textured meshes for the robot, objects, and tabletop; physically-based lighting with area lights matching the lab illumination; and domain randomization [25] over backgrounds, lighting intensity, and camera pose. Fig. 8 shows side-by-side comparisons of the real-world camera view (left) and the corresponding simulation rendering (right) across all five tasks.



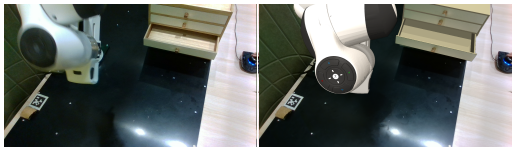
(a) Cube Lift



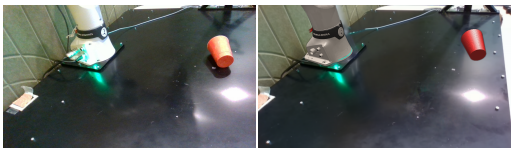
(b) Pick-and-Place



(c) Stack Cube



(d) Close Drawer



(e) Stand Cup Up

Figure 8: Realistic simulation rendering (right in each pair) vs. real-world camera view (left). The rendering is set up to support image-based RL and policy distillation as auxiliary baselines in simulation.

A.4 Algorithm Pseudocode

Algorithms 1 and 2 provide complete pseudocode for the two phases of our framework. Algorithm 1 details the residual RL training loop in simulation, including VLA action chunking, pose noise augmentation, and TD3 [38] updates. Algorithm 2 describes the zero-shot real-world deployment procedure, where a confidence-gated pose dropout replaces the stochastic dropout used during training. The pose estimator combines FoundationPose [20] for 6-DoF tracking with SAM2 [21] for instance segmentation.

A.5 Training Hyperparameters

Table 5 lists the task-specific hyperparameters used for residual TD3 [38] training. All tasks share the same network architecture (2-layer MLP with 512-unit actor and 1024-unit critic hidden layers) and optimizer (Adam). Per-task differences in learning rate, L2 regularization, discount factor, episode length, offline sampling fraction, and critic warmup accommodate the varying task dynamics.

A.6 Pose Noise and Dropout Parameters

We use $\sigma_p^{\max} = 0.005$ (5 mm) for position noise and $\sigma_q^{\max} = 0.1$ rad ($\approx 5.7°$) for orientation noise. Each component of ϵ_p is independently sampled from $\mathcal{U}(-\tilde{\sigma}_p, \tilde{\sigma}_p)$ with $\tilde{\sigma}_p \sim \mathcal{U}(0, \sigma_p^{\max})$ resampled per timestep, and ϵ_q is a small random rotation whose magnitude is similarly drawn

Algorithm 1 Object-Centric Residual RL Training

Require: Frozen $\pi_{\text{VLA}}^{\text{sim}}$, language instruction l, episode length T, chunk length H, noise parameters $\sigma_p^{\max}, \sigma_q^{\max}, \rho_{\text{drop}}$, exploration noise std $\tilde{\sigma}$, clip bound c

Initialize actor $\pi_{\text{res}}^{\text{sim}}$, critics Q_1, Q_2, target networks, replay buffer $\mathcal{B}$
for each episode **do**
  $s_0 \leftarrow$ env.reset()
  for $t = 0, 1, \ldots, T-1$ **do**
    if $t \bmod H = 0$ **then**
      $A \leftarrow \pi_{\text{VLA}}^{\text{sim}}(o_t^{\text{img}}, s_t^{\text{prop}}, l)$ ▷ Query frozen VLA, H-length chunk
    end if
    $a_t^{\text{base}} \leftarrow A[t \bmod H]$
    $\tilde{x}_{\text{obj}} \leftarrow \textsc{PoseAugment}(p_{\text{obj}}, q_{\text{obj}}, \sigma_p^{\max}, \sigma_q^{\max}, \rho_{\text{drop}})$
    $s_t \leftarrow [\, \tilde{x}_{\text{obj}},\ s_t^{\text{prop}},\ a_t^{\text{base}} \,]$
    $\delta_t \leftarrow \pi_{\text{res}}^{\text{sim}}(s_t) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ ▷ Noise on residual
    $a_t \leftarrow a_t^{\text{base}} \oplus \delta_t$
    Execute a_t in env, observe r_t, s_{t+1}
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ ▷ Store combined action
    Sample mini-batch from $\mathcal{B}$; update Q_1, Q_2 and $\pi_{\text{res}}^{\text{sim}}$ (TD3)
  end for
end for
return $\pi_{\text{res}}^{\text{sim}}$

Algorithm 2 Zero-Shot Real-World Deployment

Require: Frozen $\pi_{\text{VLA}}^{\text{real}}$, frozen $\pi_{\text{res}}^{\text{sim}}$, language instruction l, chunk length H, pose-confidence threshold τ_c

for $t = 0, 1, \ldots$ until task completion **do**
  $o_t^{\text{img}} \leftarrow$ CaptureRGB()
  if $t \bmod H = 0$ **then**
    $A \leftarrow \pi_{\text{VLA}}^{\text{real}}(o_t^{\text{img}}, s_t^{\text{prop}}, l)$ ▷ Query frozen VLA, H-length chunk
  end if
  $a_t^{\text{base}} \leftarrow A[t \bmod H]$
  $\tilde{x}_{\text{obj}}, c_t \leftarrow \text{FoundationPose}(\text{SAM2}(o_t^{\text{img}}))$ ▷ 6-DoF pose + confidence
  if $c_t < \tau_c$ **then** $\tilde{x}_{\text{obj}} \leftarrow \mathbf{0}$ ▷ Low confidence → dropout
  end if
  $s_t \leftarrow [\, \tilde{x}_{\text{obj}},\ s_t^{\text{prop}},\ a_t^{\text{base}} \,]$
  $a_t \leftarrow a_t^{\text{base}} \oplus \pi_{\text{res}}^{\text{sim}}(s_t)$
  Execute a_t on robot
end for

via $\tilde{\sigma}_q \sim \mathcal{U}(0, \sigma_q^{\max})$. These ranges match the typical depth-camera-based pose estimation error commonly reported for Intel RealSense D435 (∼2.5–5 mm at 1 m distance) and the orientation jitter of FoundationPose [20] under nominal tracking conditions. During training, pose dropout is applied with probability $\rho_{\text{drop}} = 0.1$.

Additional fixed constants. We use action chunk length $H = 16$ (the GR00T-N1.5 [8] default), exploration noise clip bound $c = 0.5$, and pose-confidence threshold $\tau_c = 0.5$ at deployment.

A.7 Sim-to-Real Behavioral Consistency

A key validation of our zero-shot transfer framework is that base VLA failure modes observed on the real robot are reproduced in simulation: because $\pi_{\text{VLA}}^{\text{sim}}$ and $\pi_{\text{VLA}}^{\text{real}}$ are trained on the same teleoperation data, they exhibit the same characteristic failures (e.g., hovering above the cube, stopping short of the target). Training the residual against these shared failures in simulation thus directly addresses the failures encountered on the real robot, consistent with the premise of residual pol-

Table 5: Task-specific training hyperparameters.

	Lift	PnP	Stack	Close	Stand
Actor hidden dim	512	512	512	512	512
Critic hidden dim	1024	1024	1024	1024	1024
Actor LR	1e-5	1e-5	3e-4	1e-5	1e-5
Critic LR	1e-4	1e-4	3e-4	1e-4	1e-4
L2 reg.	0.0	0.1	1.0	0.01	1.0
γ	0.99	0.99	0.95	0.99	0.99
Batch size	256	256	256	256	256
Offline fraction	0.5	0.5	0.75	0.3	0.5
Max episode steps	300	300	500	500	300
Critic warmup steps	1k	1k	2k	1k	1k
n-step returns	3	3	3	3	3
Target τ	0.005	0.005	0.005	0.005	0.005
Exploration noise $\tilde{\sigma}$	0.05	0.05	0.05	0.05	0.05

icy learning [14, 15]. Table 6 summarizes the shared failure modes and the corresponding residual corrections (trained in sim, deployed zero-shot to real).

Table 6: Base VLA failure modes shared between simulation and the real robot, and the residual corrections that resolve them (learned in simulation, deployed zero-shot to real).

Task	Base VLA Problem	Residual Fix
Lift, PnP, Stack	Hovers above cube, misses grasp Stops short of target position/angle Gripper collides with cube and stalls	Pushes end-effector down to the cube Moves end-effector closer to the target Adjusts approach angle
Close Drawer	Pushes at wrong angle	Redirects along drawer axis
Stand Cup Up	End-effector can't reach correct grasp pose	Guides end-effector to precise grasp position

These observations confirm the central premise of our framework: when paired sim/real VLAs share the same failure distribution, a residual trained to correct the sim failures resolves the same failures on the real robot zero-shot, without observing images at any point. Fig. 9 shows representative keyframe comparisons across all five tasks.

A.8 Strong Base VLA + Residual RL

A natural question is whether residual RL still helps when the base VLA is already strong. We investigate this by comparing two tiers of base VLA performance on the Pick-and-Place task (Table 7).

Table 7: Tier 1 vs. Tier 2: Residual RL on weak and strong base VLAs (Pick-and-Place, 20 trials).

	Method	Sim SR	Real SR
Tier 1 (Weak, GR00T-N1.5)	Base VLA	7/20	9/20
	+ Residual RL	**17/20**	**16/20**
Tier 2 (Strong, $\pi_{0.5}$)	Base VLA	18/20	**17/20**
	+ Residual RL	**19/20**	**17/20**

With a weak base (Tier 1), residual RL roughly doubles the success rate in both simulation and the real world. With a strong base (Tier 2, $\pi_{0.5}$ [7], 17/20 real), the residual maintains the same real-world success rate without degradation, while slightly improving simulation performance. This is the expected behavior: when the base already acts correctly, the residual learns to stay near zero and does not introduce unnecessary corrections.

However, residual RL remains useful for two reasons. First, VLA success rates are highly dependent on the deployment environment—a model that appears strong in one setting can become a weak base

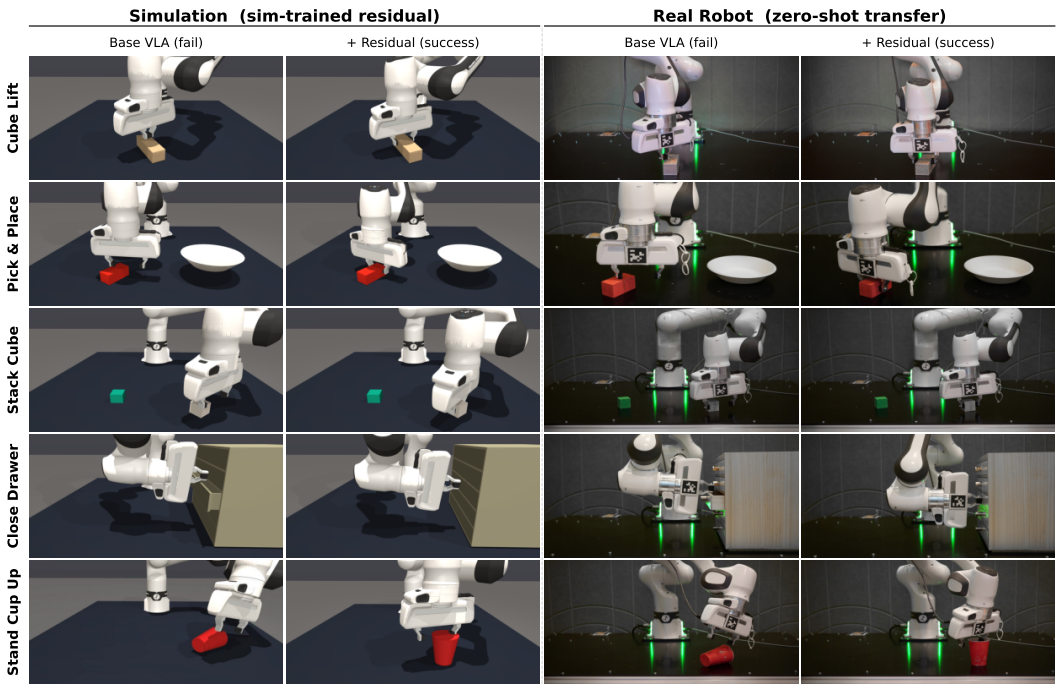


Figure 9: **Sim-to-real behavioral transfer of the object-centric residual policy.** All five tasks are shown; left two columns are simulation, right two are real-robot deployment. The residual, trained *only* in simulation, learns task-specific corrections to base VLA failure modes: downward correction during *Cube Lift* approach, lateral alignment for *Pick-and-Place*, accurate grasp positioning for *Stack Cube*, corrective pushing motion to fully close the drawer in *Close Drawer*, and re-orientation to stand the cup upright in *Stand Cup Up*. Because the residual observes object pose, a representation invariant across domains, the same corrections transfer zero-shot to the real robot.

in another, and residual RL provides a safety net in such cases. Second, when the base is weak, it is often unclear what additional demonstration data would make it stronger. Residual RL sidesteps this problem: the policy explores diverse situations in simulation and autonomously discovers failure modes that human teleoperators may not anticipate when curating fine-tuning data. Moreover, unlike real-robot data collection where performance cannot be easily evaluated during the process, sim-based residual RL allows continuous evaluation and monitoring of improvement throughout training. Finally, our sim-trained residual is complementary to recent VLAs that learn from execution experience [41]: it can be applied on top of any frozen base policy without modifying the base's training pipeline.

A.9 Emergent Behaviors from Residual RL

Beyond closing the simulation-to-real gap, residual RL also induces qualitatively new behaviors that are absent from the demonstration data used to train the base VLA [8]. Fig. 10 shows four such examples observed during real-robot deployment.

Cube Lift (*Cube Pre-Rotation*). Before grasping, the residual nudges the cube into a graspable orientation, a strategy not present in the demonstrations.

Pick-and-Place (*Cube Pre-Rotation*). The residual exhibits the same pre-rotation strategy before grasping the cube to place it in the bowl.

Stack Cube (*Corrective Push to Grasp*). When the base policy's grasp is misaligned, the residual drives the gripper toward the cube to reach a full close.

Close Drawer (*Sustained Contact Push*). The residual maintains downward contact through the late phase, avoiding the base policy's premature lift that would otherwise lose contact with the drawer.

These behaviors emerge purely from RL exploration in simulation: the policy autonomously discovers corrective strategies that human teleoperators may not anticipate when curating demonstration data, supporting the second motivation discussed in Sec. A.8.

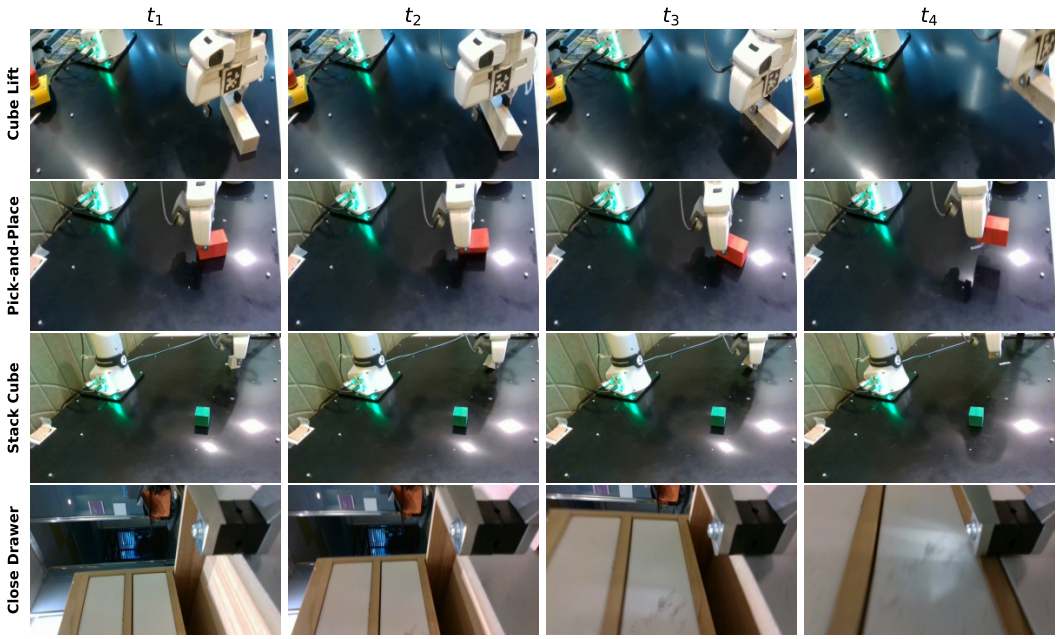


Figure 10: **Emergent behaviors from residual RL.** Each row shows four sequential keyframes (left to right in time) from a successful real-robot rollout with the residual policy. The residual discovers task-specific strategies that are absent from the demonstrations used to train the base VLA: pre-rotating the cube before grasp (*Cube Lift* and *Pick-and-Place*), corrective push toward the cube when the grasp is misaligned (*Stack Cube*), and sustained downward contact through the late phase of drawer closing (*Close Drawer*).

A.10 Object Tracking Visualization

We visualize the FoundationPose [20] object tracking results during real-robot deployment. Fig. 11 shows representative frames with the estimated 6-DoF pose visualized as a projected mesh outline of each object, for all five manipulation tasks. For Close Drawer, the policy observes the slide displacement; we visualize the closing progress using a depth-based quad detector that identifies the active drawer and estimates its slide position from the median depth within each drawer face region.

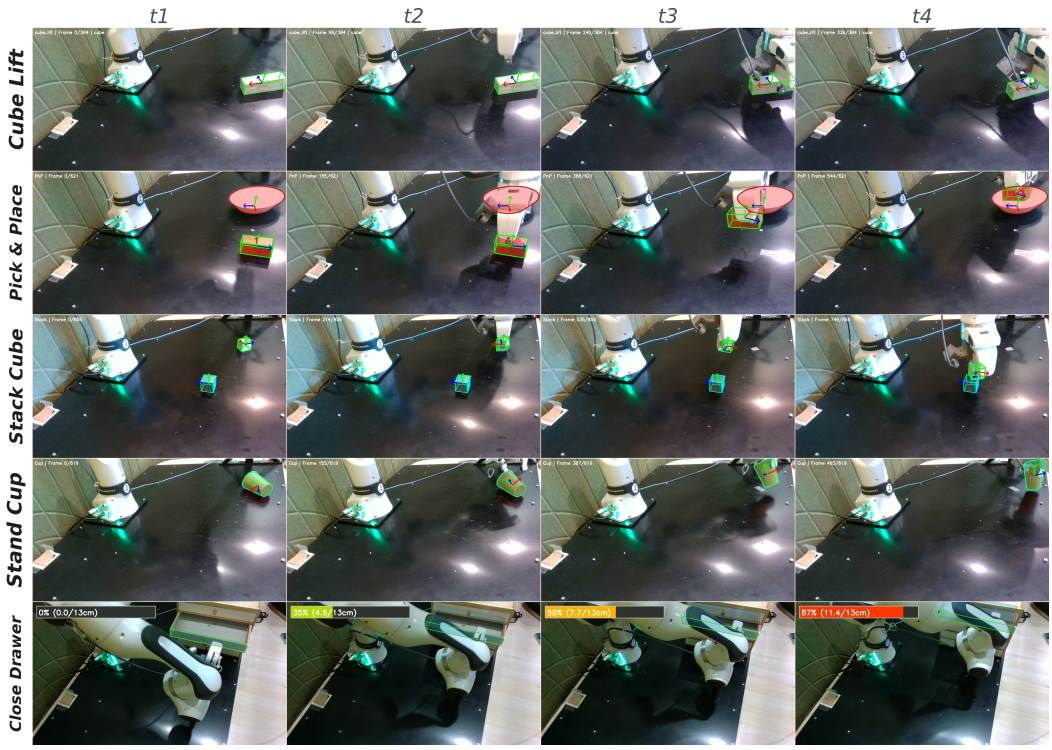


Figure 11: FoundationPose [20] pose tracking overlaid on real-robot RGB frames. Each row shows a different task (Cube Lift, Pick-and-Place, Stack Cube, Stand Cup Up, and Close Drawer) at evenly spaced timesteps during a successful episode. For each object, the estimated 6-DoF pose is visualized by projecting the object's mesh outline into the camera frame, with body-frame axes (red, green, blue) drawn at the object origin. For Close Drawer, the overlay additionally shows the closing-progress percentage on the active drawer face.

A.11 Failure Case Analysis

Fig. 12 shows three representative failure modes observed during real-robot evaluation, with the FoundationPose [20] 6-DoF estimate overlaid on each frame.

- **Pose-estimation error** (Rows 1–2). FoundationPose returns an offset or drifted pose, causing the residual to correct in the wrong direction.
- **Occlusion** (Row 3). The gripper occludes the cube as it closes, and the tracker loses the object.
- **Wrong-object detection** (Row 4). Segmentation locks onto a specular reflection instead of the target, producing a spurious pose estimate.

All three modes stem from perception failures that pass the confidence gate. Incorporating complementary verification—such as VLM-based semantic validation or multi-hypothesis pose estimation with diverse initializations—to detect plausible-but-incorrect estimates, together with fallback strategies that gracefully degrade to base VLA behavior, is a promising direction for addressing these cases.

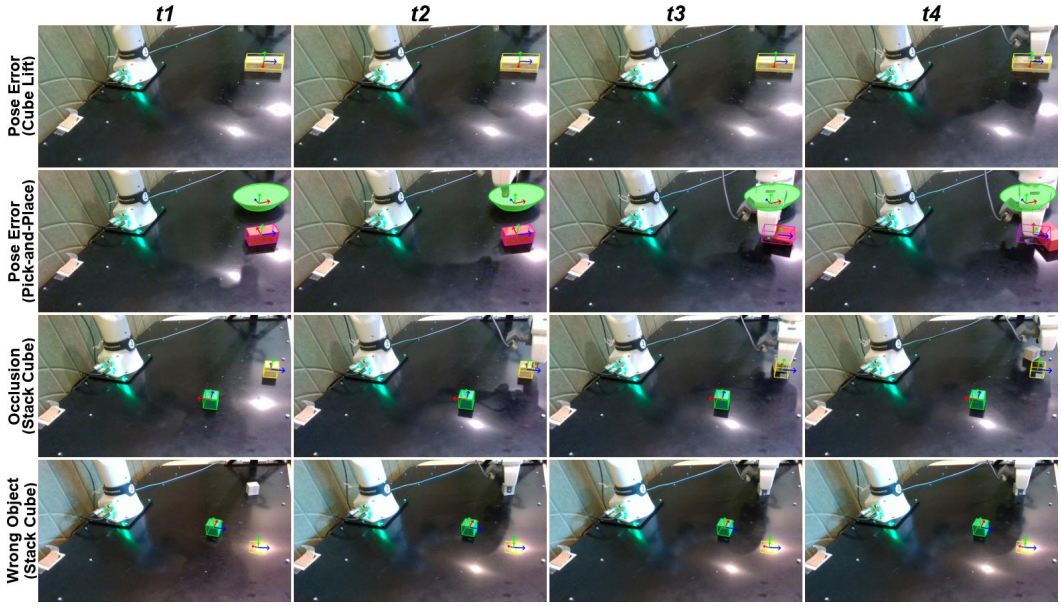


Figure 12: Representative failure modes on the real robot. Each row shows four uniformly sampled timesteps with the FoundationPose estimate overlaid. **Rows 1–2 (Pose Error)**: tracker offset on Cube Lift (Row 1) and pose drift during Pick-and-Place (Row 2). **Row 3 (Occlusion)**: gripper occludes the cube during Stack Cube grasp, causing the tracker to lose the object. **Row 4 (Wrong Object)**: segmentation locks onto a table reflection in Stack Cube.