Beyond Imitation: Reinforcement Learning–Based Sim–Real Co-Training for VLA Models

Author Names Omitted for Anonymous Review. Paper-ID 177

***Abstract*—Simulation offers a scalable and low-cost way to enrich vision-language-action (VLA) training, reducing reliance on expensive real-robot demonstrations. However, most sim-real co-training methods rely on supervised fine-tuning (SFT), which treats simulation as a static source of demonstrations and does not exploit large-scale closed-loop interaction. Consequently, real-world gains and generalization are often limited. In this paper, we propose an *RL*-based sim-real *Co*-training (RL-Co) framework that leverages interactive simulation while preserving real-world capabilities. Our method follows a generic two-stage design: we first warm-start the policy with SFT on a mixture of real and simulated demonstrations, then fine-tune it with reinforcement learning in simulation while adding an auxiliary supervised loss on real-world data to anchor the policy and mitigate catastrophic forgetting. We evaluate our framework on four real-world tabletop manipulation tasks using two representative VLA architectures, OpenVLA and $\pi_{0.5}$, and observe consistent improvements over real-only fine-tuning and SFT-based co-training, including +24% real-world success on OpenVLA and +20% on $\pi_{0.5}$. Beyond higher success rates, RL co-training yields stronger generalization to unseen task variations and substantially improved real-world data efficiency, providing a practical and scalable pathway for leveraging simulation to enhance real-robot deployment.**

I. Introduction

Building general-purpose robots that can reliably solve real-world tasks remains a central goal in robotics research. Vision-language–action (VLA) models have recently emerged as a promising foundation toward this goal, demonstrating strong performance across a wide range of embodied tasks, including robotic manipulation [7–9, 30, 35, 85] and visual navigation [6, 25–27, 37, 56, 64]. These models are typically pre-trained on large-scale real-world demonstrations [34, 53, 69], leveraging expert data to learn task-relevant perception and control behaviors. However, despite extensive pretraining, their performance often degrades significantly under novel scenes and task variations [80]. Moreover, the difficulty and cost of collecting large-scale real-robot demonstrations further constitute a major bottleneck for training VLA models exclusively on real-world data.

Simulation offers a natural alternative to alleviate this limitation. Modern simulators [24, 47, 50, 60, 66], together with large collections of open-source assets [10, 11, 19, 20], enable the construction of diverse training environments at scale. Due to the sim-to-real gap, early simulation-based robotics research primarily relied on domain randomization [1, 54, 65] to improve robustness to visual and physical discrepancies, but this approach depends on carefully hand-designed randomization schemes and scales poorly to complex, long-horizon manipulation. In recent years, real-to-sim-to-real pipelines [16, 39, 51, 71, 79] and generative modeling [59, 61, 83, 84] have substantially alleviated the sim-to-real gap by improving visual fidelity and scene diversity. However, achieving highly realistic simulation still requires accurate modeling of geometry, materials, contact dynamics, and sensing, which increases system complexity and limits scalability across tasks.

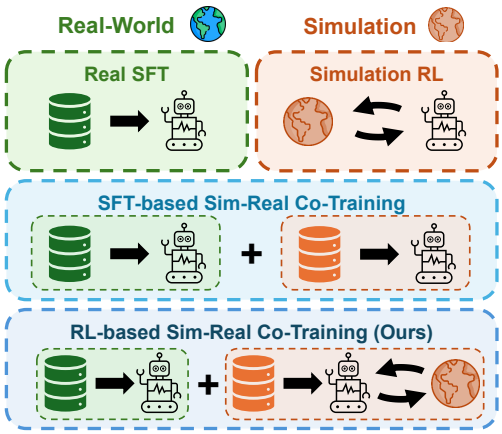

Fig. 1: **Overview of training paradigms combining real-world and simulated data.** VLA models are commonly trained via supervised fine-tuning (SFT) on real-world demonstrations, or via reinforcement learning (RL) in simulation followed by sim-to-real transfer. Other approaches adopt SFT-based sim–real co-training by mixing real and simulated demonstrations. In contrast, we propose an RL-based sim–real co-training (RL-Co) framework, which initializes the model with sim–real SFT and subsequently performs RL in simulation while using real-world SFT as a regularization signal.

Beyond direct sim-to-real transfer, several recent studies [2, 7, 16, 23, 46, 51, 70, 73, 75] have explored sim–real co-training paradigms that jointly leverage simulated and real-world data. By leveraging scalable simulation data, these approaches consistently outperform policies trained solely on real-world demonstrations. Notably, co-training has been shown to remain effective even when the simulated visual appearance differs substantially from the real world [70], or when simulation tasks are only loosely related to the target real-world task [46]. Despite their empirical success, existing sim–real co-training methods largely remain within the supervised learning paradigm, using simulation primarily as a source of static demonstration data. This design fails to fully exploit a key advantage of simulation: its ability to

support scalable, closed-loop interaction with the policy.

Meanwhile, prior work has pointed out that VLA models trained purely with supervised fine-tuning (SFT) for behavior cloning are inherently susceptible to compounding errors under distribution shift, which can accumulate over time and limit robust performance [55]. To overcome the limitations, recent work [38, 42, 43, 81, 82] has explored reinforcement learning (RL) as an alternative post-training paradigm for VLA models. By fine-tuning VLA policies through interactive learning, these methods achieve higher task success rates and significantly improved generalization to unseen scenarios compared to SFT-based approaches. However, although these methods perform well in simulation, their real-world deployment typically depends on zero-shot sim-to-real transfer with domain randomization, frequently leading to significant performance drops on real robots.

In this work, we propose an *RL*-based sim-real *Co*-training (RL-Co) framework for VLA models that goes beyond static demonstrations by leveraging interactive simulation, while preserving real-world capabilities. Our framework adopts a simple two-stage design. We first initialize the policy via supervised co-training on a mixture of real-world and simulated demonstrations, transferring task-relevant real-world knowledge while establishing a strong simulation prior. We then further optimize the policy with reinforcement learning in simulation. To preserve real-world capabilities and mitigate catastrophic forgetting, we add an auxiliary supervised loss on real-world demonstrations during simulation RL to anchor the policy.

To demonstrate the efficacy of our RL-Co framework, we conduct extensive experiments on four real-world tabletop manipulation tasks with two representative VLA models, OpenVLA [35] and $\pi_{0.5}$ [30]. Across all tasks and models, RL-Co consistently outperforms real-only fine-tuning and SFT-based sim–real co-training, yielding substantial improvements in real-world success rates. Beyond raw performance gains, we find that our approach exhibits significantly better generalization to unseen task variations and is markedly more stable with respect to hyperparameter choices than SFT-based co-training. Moreover, by effectively leveraging large-scale simulated interaction, our method substantially reduces the amount of required real-world demonstration data, demonstrating a more data-efficient and scalable pathway for deploying VLA models on real robots.

II. Related Works

A. Vision-Language-Action Models for Manipulation Tasks

Vision-Language-Action (VLA) models have revolutionized robotic control by integrating visual perception and linguistic reasoning into a foundation model [8, 9, 30, 35, 63, 85]. Built upon the success of Large Language Models or Vision-Language Models [3, 5, 62, 68, 72], these systems are typically pretrained on massive datasets of internet-scale images [12, 15, 57] and robotic demonstrations [22, 53]. This extensive pretraining endows VLAs with remarkable generalization capabilities, allowing them to follow natural language instructions and perform diverse manipulation tasks across different embodiments.

B. Fine-Tuning VLA Models via Reinforcement Learning

Post-training is crucial for adapting pretrained VLA models to downstream manipulation tasks. Most existing methods rely on Supervised Fine-Tuning (SFT), which effectively aligns models with target distributions using limited demonstrations [28, 31, 36]. However, SFT suffers from covariate shift, where compounding errors cause policies to deviate from expert trajectories [43, 52, 55].

To address this limitation, recent works incorporate reinforcement learning (RL) into the post-training stage, enabling policies to improve through interaction and trial-and-error. Depending on the VLA architecture, diverse RL-based strategies have been explored [4, 14, 29, 38, 40, 42, 43, 45, 76, 78, 81]. For instance, Li et al. [38] exploit temperature sampling in OpenVLA [35] to support PPO-based fine-tuning [58], while Zhang et al. [81] introduce stochasticity into flow matching denoising [41] to enable effective exploration.

Despite these advances, most RL-based VLA training is conducted in simulation for safety and efficiency, requiring sophisticated sim-to-real transfer or extensive domain randomization. Direct real-world RL avoids this gap [4, 29, 32, 40, 45] but is limited by high cost, safety risks, and slow data collection [21]. In contrast, our method bridges simulated RL and real-world data constraints, achieving efficient policy improvement without heavy sim-to-real engineering.

C. Sim-to-Real Transfer and Sim-Real Co-Training

Simulation provides a safe and scalable platform for robotic learning, yet the sim-to-real gap remains a fundamental challenge. A common strategy is to build high-fidelity digital twins that reduce this gap through accurate visual and physical modeling [13, 33, 67, 74]. However, such replicas are expensive to construct and still struggle to capture the full complexity of real-world environments. Alternatively, Domain Randomization (DR) improves robustness by heavily randomizing visual and physical parameters during simulation [1, 13, 49, 54, 65], but often requires extensive training and careful manual tuning to avoid overly conservative policies.

Beyond direct transfer, recent work has shifted toward sim-real co-training, jointly optimizing policies with both simulated and real-world data [2, 7, 16, 23, 46, 51, 70, 73, 75]. Some methods reduce the domain gap by learning invariant representations shared across simulation and reality [17, 73, 75], while others primarily leverage simulation as large-scale data augmentation to improve generalization, even when visual fidelity or task alignment is limited [16, 46, 48, 50, 51, 77].

Despite these advances, most co-training approaches treat simulation as a static source of trajectories, overlooking its interactive nature. Our method builds on the data augmentation paradigm while incorporating reinforcement learning into the co-training loop, enabling active exploration in simulation and grounding the policy with real-world data.

III. Preliminaries

A. Problem Formulation

For each real-world robotic manipulation task T_{real}, we construct a corresponding digital-twin simulation environment, resulting in a simulation task T_{sim} that serves as a digital-twin of the real task [18]. The simulation environment is designed to closely mirror the real-world setup while allowing scalable data collection through interaction.

We model both the real-world task and its simulated counterpart as Partially Observable Markov Decision Processes (POMDPs), denoted by the tuple

$$\mathcal{M}_\Omega = \langle \mathcal{S}_\Omega, \mathcal{A}, \mathcal{P}_\Omega, \mathcal{R}, \mathcal{O}_\Omega, \mathcal{L}, P(s_0), \gamma \rangle, \tag{1}$$

where $\Omega \in \{\text{real}, \text{sim}\}$ indicates whether the process corresponds to the real-world or simulation task.

Following the formulation in [46], we define each component as follows:

- $\mathcal{S}_\Omega$ and $\mathcal{O}_\Omega$ denote the state space of the robot–environment system and the observation space induced by onboard sensors, respectively. While the real and simulated tasks operate in different environments, they share the same robot embodiment and sensing modalities.
- $\mathcal{A}$ is the robot action space. Both tasks adopt an identical control interface and action parameterization.
- $\mathcal{P}_\Omega$ represents the state transition dynamics, where $s_{t+1} \sim \mathcal{P}_\Omega(\cdot \mid s_t, a_t)$. Due to the inherent difficulty of perfectly modeling real-world physics, the transition dynamics in simulation may exhibit slight discrepancies from those in the real environment.
- $\mathcal{L}$ denotes the natural language instruction specifying the task goal. For corresponding real and simulated tasks, the language instruction remains identical.
- $\mathcal{R}$ is the reward function, defined as $\mathcal{R}(s, l)$, which evaluates task progress based on the current state and the given language instruction.
- $P(s_0)$ is the distribution over initial states, from which $s_0 \sim P(s_0)$ is sampled. The real and simulated tasks share the same initial state distribution.
- $\gamma \in (0, 1)$ is the discount factor.

Under this formulation, we define a vision-language-action (VLA) policy π_θ that conditions on the most recent H observations $o_\Omega^{t-H+1:t}$ and the language instruction l to predict a sequence of future actions over a horizon of length h:

$$a_{t:t+h-1} \sim \pi_\theta\big(a_{t:t+h-1} \mid o_\Omega^{t-H+1:t}, l\big). \tag{2}$$

B. Fine-Tuning on Vision-Language-Action Models

We consider post-training of vision-language-action (VLA) models under both supervised and reinforcement learning paradigms. Given a pre-trained VLA policy π_θ, fine-tuning aims to adapt the policy to a specific manipulation task by leveraging either expert demonstrations or online interaction with the environment.

1) Supervised Fine-Tuning (SFT): Given an expert-collected demonstration dataset $\mathcal{D}_T = \{(\tau^{(i)}, l^{(i)})\}_{i=1}^N$, each trajectory $\tau^{(i)} = \{(o_j^{(i)}, a_j^{(i)})\}_{j=1}^{K_i}$ consists of paired observations and actions, and $l^{(i)}$ denotes the corresponding natural language instruction. Here, N is the total number of trajectories and K_i is the length of the i-th trajectory.

Supervised fine-tuning optimizes the VLA policy π_θ by minimizing the discrepancy between predicted and expert actions:

$$L_{\text{SFT}}(\theta) = \mathop{\mathbb{E}}_{\substack{(\tau,l)\sim D_T \\ t\sim \text{Unif}(\{1,\ldots,K_\tau\})}} \Big[\ell_{\text{SFT}}(\hat{a}_{t:t+h-1},\, a_{t:t+h-1})\Big], \tag{3}$$

where

$$\hat{a}_{t:t+h-1}^{(i)} = \pi_\theta\big(o_{t-H+1:t}^{(i)},\, l^{(i)}\big) \tag{4}$$

denotes the predicted action chunk of horizon h, and

$$a_{t:t+h-1}^{(i)} = \{a_t^{(i)}, a_{t+1}^{(i)}, \ldots, a_{t+h-1}^{(i)}\} \tag{5}$$

is the corresponding expert action sequence.

The loss function ℓ_{SFT} depends on the specific VLA architecture and action representation. Common choices include next-token prediction losses [35], L_1 regression losses for continuous actions [36], and diffusion-based denoising objectives [8].

2) Reinforcement Learning (RL) Fine-Tuning: Reinforcement learning fine-tuning seeks to further optimize the policy through interaction with the environment by maximizing the expected discounted return:

$$\pi^* = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta, \mathcal{P}} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, l)\right], \tag{6}$$

where actions are sampled from the VLA policy $a_t \sim \pi_\theta(\cdot \mid o_t, l)$ and state transitions follow $s_{t+1} \sim \mathcal{P}(s_t, a_t)$.

Due to differences in action representations and generative mechanisms, the concrete realization of RL fine-tuning varies across VLA architectures. Nevertheless, existing RL fine-tuning approaches share a common structure: an iterative loop of environment interaction for trajectory collection, followed by policy updates guided by reward feedback. Our method builds upon this general framework and introduces an additional supervised fine-tuning objective on real-world data during the policy update phase, which is compatible with a wide range of RL fine-tuning strategies.

C. SFT-based Co-Training

Given a real-world manipulation task T_{real} and its corresponding digital-twin simulation task T_{sim}, we assume access to expert demonstration datasets $\mathcal{D}_{\text{real}}$ and $\mathcal{D}_{\text{sim}}$, collected in the real and simulated environments, respectively. A straightforward approach to leverage both sources of supervision is to jointly fine-tune the VLA policy using a mixture of real and simulated demonstrations.

Specifically, supervised co-training is formulated as minimizing a weighted combination of the SFT losses over the two datasets:

$$\mathcal{L}_{\text{SFT}}(\theta) = \alpha\, \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_{\text{sim}}) + (1-\alpha)\, \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_{\text{real}}), \tag{7}$$

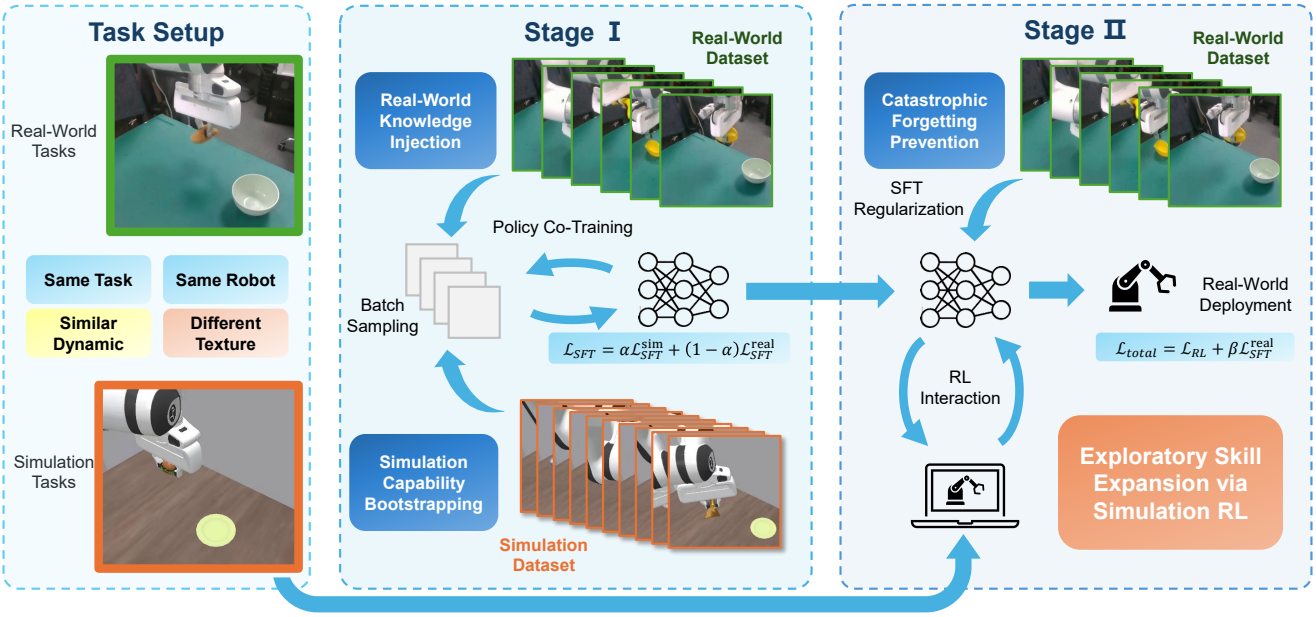

Fig. 2: **Overview of the proposed two-stage sim-real co-training framework.** We establish a digital-twin setup where T_{sim} serves as a digital cousin to T_{real} despite visual discrepancies. In **Stage I**, we initialize the VLA policy by supervising it on a mixture of real and simulated data (ratio α). This rapidly injects real-world knowledge and prepares the policy for simulation interaction. In **Stage II**, we perform RL fine-tuning in the simulator to explore and improve performance, simultaneously employing a real-world SFT loss as a regularizer to prevent the forgetting of real-world behaviors.

where $\alpha \in [0, 1]$ controls the relative contribution of simulated data during training.

Following Maddukuri et al. [46], this objective can be equivalently implemented by sampling training trajectories from the simulation dataset with probability α, and from the real-world dataset with probability $1 - \alpha$.

This SFT-based co-training strategy is a strong and widely used baseline for sim-to-real transfer. However, due to the imitation learning objective, it is constrained by the quality of data curation and sim-to-real gaps and cannot explicitly leverage reward feedback or online interaction. These limitations motivate the reinforcement learning–based co-training approach introduced next.

IV. Method

In this section, we present our *RL*-based sim-real *Co*-training (RL-Co) framework. An overview of the proposed method is shown in Fig. 2. Our approach consists of two successive stages. In Stage I, we initialize the policy via supervised co-training on both real-world and simulated demonstrations. In Stage II, we further improve the policy through reinforcement learning in simulation, while explicitly preserving real-world capabilities via an auxiliary supervised objective.

A. Stage I: SFT Co-Training for Policy Initialization

Starting from a pre-trained VLA policy π_θ that has not been adapted to our target tasks, the first stage aims to initialize the policy using both real-world and simulated demonstrations. Specifically, we apply supervised fine-tuning co-training using the real-world dataset $\mathcal{D}_{\text{real}}$ and the simulation dataset $\mathcal{D}_{\text{sim}}$, as described in Section III-C.

This stage serves two critical purposes. First, it enables the policy to rapidly incorporate task-specific real-world knowledge, which is essential for downstream deployment. Second, by simultaneously learning from simulated demonstrations, the policy acquires a reasonable level of competence in the simulation environment, ensuring a non-trivial task success rate and thus providing a suitable initialization for reinforcement learning.

These two properties motivate our choice of SFT co-training as the first stage of the proposed framework, serving as an initialization step before RL co-training. We defer a detailed analysis of its contribution to Section V-C.

B. Stage II: Sim-Real Co-Training with Real-Regularized RL

While Stage I equips the policy with both real-world and simulated capabilities, its optimization is limited to imitation objectives. In Stage II, we seek to further expand the policy's competence through online interaction in simulation, while preventing the degradation of real-world performance.

To achieve this, we introduce an auxiliary supervised fine-tuning objective on real-world data into the reinforcement learning fine-tuning process. During RL training in simulation, each policy update is typically driven by a reinforcement learning loss $\mathcal{L}_{\text{RL}}$, which encourages exploration and maximization of task rewards. We augment this objective with an additional SFT loss computed on $\mathcal{D}_{\text{real}}$, resulting in the following combined optimization objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}} + \beta\, \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_{\text{real}}), \tag{8}$$

where β is a weighting coefficient that balances reinforcement learning updates and preservation of real-world knowledge.

Intuitively, the RL term enables the policy to leverage large-scale simulated interaction to explore diverse behaviors and improve task performance, while the real-world supervision term acts as a regularizer that anchors the policy to real-world demonstrations, mitigating catastrophic forgetting during RL fine-tuning. This simple yet effective modification is compatible with a wide range of RL fine-tuning algorithms and forms the core of our RL-Co framework.

V. Experiments

In this section, we empirically evaluate the proposed RL-Co framework and aim to answer the following questions:

- Does RL-Co improve real-world performance compared to training with real-world data only or SFT-based sim–real co-training?
- How do the individual components in our two-stage framework contribute to the final performance?
- To what extent can our method reduce the amount of required real-world demonstration data?

To address these questions, we first compare our method against real-world-only SFT and SFT-based sim–real co-training across a suite of manipulation tasks, demonstrating consistent improvements in real-world deployment performance (Section V-B). We further conduct targeted case studies to analyze the advantages of RL-Co in terms of generalization, and systematically explore the impact of different co-training ratios α and SFT regularization weights β. These analyses show that incorporating RL effectively expands the capability boundary of VLA models beyond what can be achieved with SFT alone.

Next, we perform ablation studies to systematically examine the role of each component in our two-stage pipeline, validating the necessity of both the SFT initialization and the real-world–regularized RL fine-tuning stage (Section V-C).

Finally, we investigate the data efficiency of our approach by comparing it with baseline methods under varying amounts of real-world demonstrations. The results highlight the potential of our method to substantially reduce real-world data requirements while maintaining strong performance (Section V-D).

A. Experimental Setting

1) Environmental Setting: To evaluate the proposed model, we design four tabletop manipulation tasks that require diverse perception, language grounding, and control skills. An overview of the real-world and simulated environments is shown in Fig. 3.

- `Pick and Place`. The robot is required to grasp objects of varying shapes from the table and place them into a target container.
- `Push Cube via Instruction`. Three cubes with different colors are placed on the table, and the robot must push the correct cube according to a natural language instruction.
- `Open Drawer`. The robot is tasked with opening a closed drawer placed on the table.

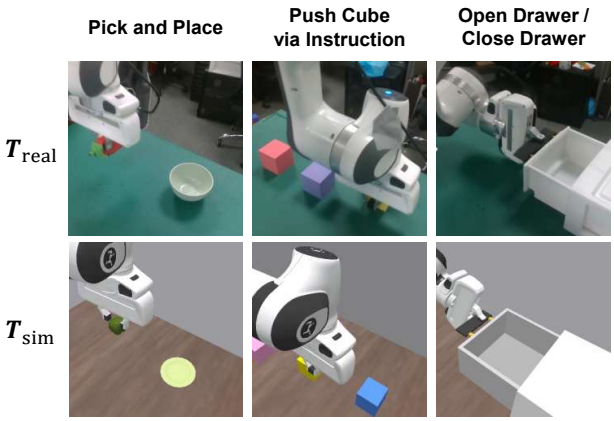

Fig. 3: **Visualization of our tabletop manipulation tasks.** The top row shows images captured by a third-person camera in the real-world setup, while the bottom row presents the corresponding simulated views. Both real and simulated images are sampled from the task execution.

- `Close Drawer`. The robot is required to close an opened drawer on the table.

We construct the simulation environments using ManiSkill [60], matching the real-world setup in terms of camera viewpoints and scene layout. Rather than pursuing photorealistic simulation or advanced visual reconstruction, we only model the essential object meshes and geometry required for task execution, without replicating low-level visual properties such as materials or lighting.

Both simulation and real-world experiments use a Franka Emika Panda robot with 7-DoF end-effector delta control. In the real world, observations are captured by an RGB camera using RGB inputs only.

Across all tasks, the camera pose and robot initial configuration are fixed, while object positions are randomly sampled within a predefined region. For fair comparison, all methods are evaluated on the same set of independently sampled initial states. Each setting is evaluated twice, and performance is reported as task success rate.

2) Dataset Generation: **Real-World Demonstrations.** For all four tasks, we collect expert demonstrations via human teleoperation using a 3D SpaceMouse. Each trajectory starts from the same initial conditions as evaluation: the robot is reset to a fixed configuration, while task-relevant objects are randomly placed on the table. Expert actions are recorded as end-effector delta control commands. For each task, we collect 20–50 successful trajectories, forming the real-world dataset $\mathcal{D}_{\text{real}}$.

Simulation Dataset Generation. To scale up training data in simulation, we adopt MimicGen [48] to generate large numbers of successful trajectories. Instead of collecting teleoperated demonstrations directly in simulation, we replay real-world expert trajectories in ManiSkill and use them as seed trajectories for data generation, thereby grounding the simulation data in real-world behaviors.

We implement the MimicGen pipeline within ManiSkill and

VLA Model	Experiment Setting	`Pick and Place`	`Push Cube`	`Open Drawer`	`Close Drawer`	`Avg`
OpenVLA	Real-Only Training	6.3 $\pm$ 0.0	20.0 $\pm$ 13.3	0.0 $\pm$ 0.0	10.0 $\pm$ 10.0	16.5 $\pm$ 13.3
	SFT Co-Training	23.4 $\pm$ 4.7	51.7 $\pm$ 5.0	0.0 $\pm$ 0.0	85.0 $\pm$ 5.0	40.0 $\pm$ 3.7
	RL-Co (Ours)	**58.8** $\pm$ 10.0	**68.3** $\pm$ 11.7	**35.0** $\pm$ 15.0	**95.0** $\pm$ 5.0	**64.0** $\pm$ 0.7
$\pi_{0.5}$	Real-Only Training	71.9 $\pm$ 9.4	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	35.0 $\pm$ 15.0	26.7 $\pm$ 1.4
	SFT Co-Training	68.8 $\pm$ 9.4	10.0 $\pm$ 3.3	10.0 $\pm$ 0.0	95.0 $\pm$ 5.0	45.9 $\pm$ 4.4
	RL-Co (Ours)	**81.3** $\pm$ 9.4	**18.4** $\pm$ 1.7	**65.0** $\pm$ 5.0	**100.0** $\pm$ 0.0	**66.2** $\pm$ 4.0

TABLE I: **Comparison of real-world success rates under different training paradigms.** We compare our RL-Co approach with real-only SFT and SFT co-training across four tabletop manipulation tasks, evaluated on both OpenVLA and $\pi_{0.5}$. Results are reported in terms of success rate (SR, %). All values are presented as mean $\pm$ standard deviation.

Experiment Setting	In-Distribution	Unseen Objects	Unseen States
Real-Only	71.9	25.0 (46.9$\downarrow$)	40.0 (31.95$\downarrow$)
SFT Co-Training	68.8	31.3 (37.5$\downarrow$)	55.0 (13.8$\downarrow$)
RL-Co (Ours)	**81.3**	**56.3** (25.0$\downarrow$)	**70.0** (11.3$\downarrow$)

TABLE II: **Comparison of generalization under unseen settings.** We evaluate all $\pi_{0.5}$ models on the `Pick and Place` task under out-of-distribution conditions, including unseen objects and unseen states. We report the success rate (SR, %) as well as the relative performance drop compared to the in-distribution setting.

introduce a minor modification: for each seed trajectory, we retain only task-relevant key stages and remove long segments of free-space end-effector motion. This pruning encourages smoother and more efficient generated trajectories. For each task, we generate 1,000 successful trajectories, which together form the simulation dataset $\mathcal{D}_{\text{sim}}$.

3) Implementation: To validate the generality of RL-Co across different model families, we implement our method on two representative vision-language-action policies: the next-token prediction-based OpenVLA [35] and the flow-matching-based $\pi_{0.5}$ model [30]. In the SFT co-training stage, real-world and simulation datasets are directly mixed, and training is conducted using the official open-source implementations provided by each model.

For OpenVLA, we follow the training protocol of Liu et al. [43] and extend their open-source codebase by incorporating the proposed real-world regularization loss during the RL optimization stage. For the $\pi_{0.5}$ model, we adopt ReinFlow [81] as the RL training algorithm. To improve training efficiency and scalability, we use RLinf [76] as the underlying training framework and integrate the real-world regularization term into the overall RL objective.

For all RL trainings, we fix the total number of environment interaction steps for each model–task pair and ensure that the RL objective in simulation is trained until convergence.

B. Main Results

To evaluate the effectiveness of the RL-Co, we compare our method against two baselines: supervised fine-tuning using real-world demonstrations only, and SFT-based sim–real co-training. The quantitative results are reported in Table I.

We first observe that fine-tuning VLA models with only a small number of real-world demonstrations leads to poor performance across most tasks. This limitation is particularly pronounced for OpenVLA, which achieves success rates below 20% across all four environments. The $\pi_{0.5}$ model performs better on the relatively simple `Pick and Place` task, but still struggles in more challenging settings. Introducing simulated demonstrations via SFT-based sim–real co-training partially alleviates these issues, yielding clear gains on simpler tasks (e.g., `Close Drawer`), but providing only limited improvement on more complex tasks. Moreover, when real-only fine-tuning already achieves strong performance, SFT-based co-training can occasionally lead to slight degradation, suggesting that purely imitation-based co-training does not consistently translate simulated data into effective real-world improvements.

In contrast, RL-Co consistently yields substantially higher real-world success rates across all task and model combinations, with three settings showing improvements of more than 35%. These results demonstrate that incorporating reinforcement learning enables simulated interaction to enhance task execution capability more effectively than both real-only fine-tuning and SFT-based sim–real co-training.

Improvement of Generalization by RL-Co. To further understand how reinforcement learning contributes to improved policy performance, we conduct an additional experiment to evaluate generalization under distribution shifts. Our motivation is inspired by Liu et al. [43], which suggests that RL fine-tuning can endow policies with stronger generalization capability than supervised fine-tuning alone. We focus on the `Pick and Place` task and evaluate the performance of the $\pi_{0.5}$ policy under two unseen settings: (i) *Unseen Objects*, where the manipulated objects are replaced with novel categories that differ from those used during training; and (ii) *Unseen States*, where the robot initial pose is perturbed in ways not encountered during either SFT or RL training. The results are summarized in Table II.

Under the original setting, all three methods achieve comparable success rates, with RL-Co showing a slight advantage. However, under unseen settings, real-only fine-tuning degrades sharply, with success rates dropping by more than 45% for unseen objects and 30% for unseen states, indicating limited robustness to changes in object properties and initial conditions. SFT-based sim–real co-training improves generalization over real-only training, achieving higher success rates in both unseen settings, suggesting that incorporating simulation data

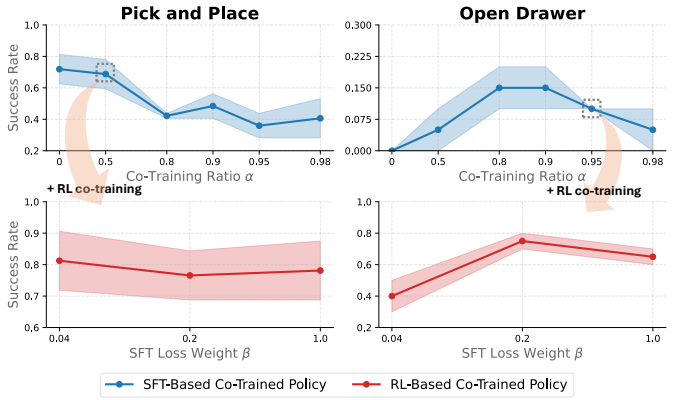

Fig. 4: **Analysis of the co-training ratio (α) and regularization weight (β).** We vary the co-training ratio α and evaluate the resulting performance on the `Pick and Place` and `Open Drawer` tasks. In addition, we fix $\alpha = 0.5$ for `Pick and Place` and $\alpha = 0.95$ for `Open Drawer`, reporting RL co-training results under different regularization weights β. Performance is measured by success rate, with shaded regions indicating standard deviation.

enhances robustness beyond the training distribution. Nevertheless, SFT-based co-training still exhibits substantial performance degradation, particularly for unseen object categories, where success rates drop by over 35%.

In contrast, RL-Co demonstrates significantly stronger generalization, substantially outperforming both baselines under distribution shifts, with markedly smaller performance degradation in both unseen-object and unseen-state evaluations. These results indicate that incorporating reinforcement learning enables the policy to acquire more robust and transferable behaviors beyond what can be achieved through supervised co-training alone.

Impact of Different SFT Co-Training Ratios α and Real-World Regularization Weights β. We further analyze the impact of two key hyperparameters in our framework: the data mixture ratio α used in the SFT-based co-training stage, and the weight β of the supervised regularization loss applied during the RL fine-tuning stage. Experiments are conducted on the `Pick and Place` and `Open Drawer` tasks using the $\pi_{0.5}$ model. Specifically, we vary α during the SFT-based co-training stage and, then select one resulting model to perform RL co-training with different values of β.

As shown in Fig. 4, the mixture ratio α has a significant impact on the performance of SFT-based co-training. For the `Pick and Place` task, strong performance can already be achieved using real-world data only, and increasing the proportion of simulated data during co-training leads to degraded real-world performance. In contrast, for the more challenging `Open Drawer` task, neither a very small nor an excessively large simulation ratio yields optimal results, suggesting that an intermediate range of α provides a better balance between real and simulated supervision.

Similarly, the regularization weight β also has a substantial

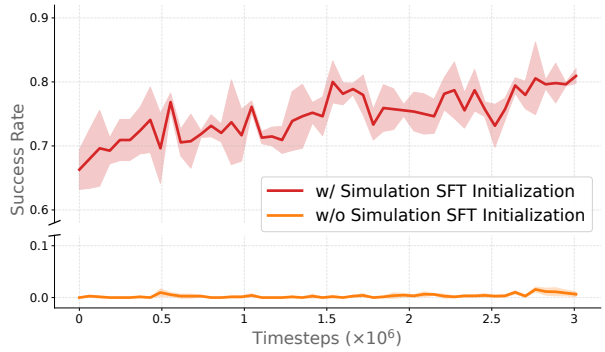

Fig. 5: **Ablation study on simulation SFT initialization.** We report the simulation success rate during RL training for models trained with and without simulation SFT initialization. Each RL training process is run with three independent random seeds, and results are presented as the mean success rate with shaded regions indicating the standard deviation.

impact on the final performance. Notably, across the three evaluated values of β, RL co-training consistently yields large performance improvements over the corresponding SFT-co-trained models, with success rates exceeding those of all SFT-only models trained under different α settings. These results indicate that reinforcement learning effectively extends the performance limits of SFT-based co-training.

C. Ablation Study

We conduct ablation studies to analyze the contribution of each component in RL-Co. Specifically, we focus on two questions: (i) how simulation data in Stage I affects RL optimization, and (ii) what roles real-world SFT plays in Stage I and Stage II.

1) Effect of Simulation Data in Stage I: To evaluate the necessity of simulated data in Stage I, we directly perform RL co-training starting from a policy trained only with real-world demonstrations. Fig. 5 compares its success rate in simulation with a policy initialized via full SFT co-training. Without simulated SFT initialization, the policy exhibits extremely poor sample efficiency and maintains a near-trivial success rate even after over three million interaction steps, whereas SFT co-training with simulated demonstrations provides a much stronger initialization and enables efficient RL optimization. This result demonstrates that simulation data in Stage I is essential for making subsequent RL-based co-training effective.

2) Role of Real-World Supervision in Two Stages: We further investigate the role of real-world supervision in each stage by removing it from Stage I and Stage II respectively. Fig. 6 reports the real-world success rates of the `Pick and Place` task using the $\pi_{0.5}$ model under all ablation settings. When the real-world SFT regularization is removed from Stage II, the success rate drops significantly from 81.38% to 40.25%, indicating that without explicit real-data anchoring, the policy suffers from catastrophic forgetting during RL optimization in simulation, even though its simulated performance continues to improve. Similarly, when real-world SFT is removed from

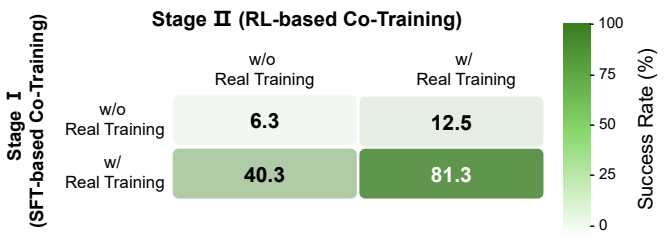

Fig. 6: **Ablation study on real-world supervision.** We ablate real-world supervised training in Stage I and Stage II separately and report the resulting real-world success rates.

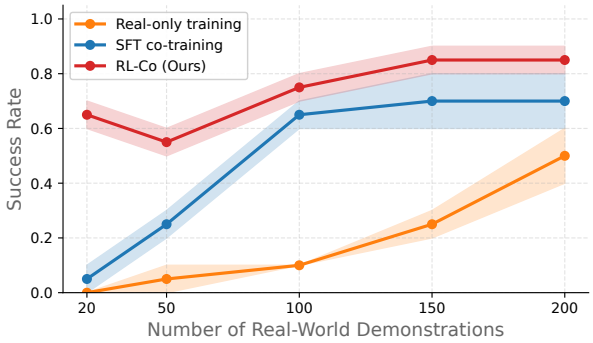

Fig. 7: **Effect of the number of real-world demonstrations.** We vary the amount of real-world demonstrations for the `Open Drawer` task and evaluate all training paradigms using the $\pi_{0.5}$ model. Performance is reported in terms of success rate, with shaded regions indicating the standard deviation.

Stage I, the final performance further degrades to 12.5%. This observation highlights the fact that, compared to reinforcement learning, SFT is substantially more data-efficient in exploiting limited real-world demonstrations. Since RL requires extensive interaction with the simulator, its learning efficiency is insufficient for acquiring real-world skills from scratch, which also explains why the real-world SFT term in Stage II mainly serves as a regularizer to preserve existing knowledge rather than enabling effective real-world learning. Finally, when real-world supervision is removed from both stages, performance collapses to 6.25%, demonstrating that directly zero-shot transferring policies trained purely in simulators with limited visual fidelity remains highly challenging.

D. Data Efficiency

As shown in Section V-B, RL-Co outperforms both real-only training and SFT-based co-training under the same amount of real-world supervision. We further investigate the data efficiency of our approach by analyzing how much real-world data can be saved compared to these baselines. To this end, we conduct a data-efficiency experiment on the `Open Drawer` task, which is representative of contact-rich manipulation. Starting from the original real-world dataset, we extend the expert demonstrations to 200 trajectories and evaluate how performance scales with varying amounts of real-world data. Specifically, we measure the success rates of real-only SFT, SFT co-training, and RL-Co as the number of real-world demonstrations increases, and report the results in Fig. 7.

As expected, all methods benefit from additional real-world demonstrations, exhibiting consistent performance improvements as the dataset size increases. Moreover, with the assistance of simulated data, SFT co-training improves more rapidly than real-only training, achieving a success rate of 65% with only 100 real-world demonstrations, which already surpasses the performance of real-only training using the full set of 200 demonstrations. However, despite this steady improvement, both baselines remain substantially inferior to RL-Co: even when trained with 200 real-world demonstrations, their performance is lower than or comparable to that of our method trained with only 20 real-world demonstrations. These results demonstrate a clear and pronounced advantage of RL-Co in terms of real-world data efficiency under the evaluated settings.

VI. Conclusion

This paper proposes RL-Co, an RL-based sim–real co-training framework for vision-language-action (VLA) models. RL-Co addresses a key limitation of prior sim–real co-training methods that rely primarily on supervised fine-tuning. The framework follows a general two-stage pipeline and is compatible with a wide range of learning algorithms and VLA architectures. We first initialize the policy via supervised fine-tuning on a mixture of simulated and real demonstrations. We then optimize the policy with reinforcement learning in simulation, while applying an auxiliary supervised loss on real-world data to preserve real-world behaviors. By incorporating online interaction and reward feedback, RL-Co goes beyond static imitation, reduces compounding errors, and mitigates catastrophic forgetting that can arise in purely supervised training or simulation-only RL.

Extensive real-world experiments across tasks and popular VLA models validate the effectiveness of our approach. RL-Co consistently outperforms real-only fine-tuning and SFT-based co-training, yielding substantial gains in real-world success rates, stronger robustness to distribution shifts, and markedly improved data efficiency. These results suggest that reinforcement learning can better realize the value of simulation in co-training, pushing performance beyond what imitation objectives alone can achieve.

Limitations. Despite these promising results, our study has several limitations. First, we evaluate only tabletop manipulation on a single robot embodiment, and we do not explore co-training across heterogeneous sim–real settings. Second, while RL-Co improves real-world success, performance remains below 100%, and we do not yet incorporate real-world RL, which may further improve robustness. Future work will extend the framework to more diverse tasks, longer-horizon manipulation, additional robot embodiments, and more efficient sim–real RL co-training with improved sim-to-real alignment.

REFERENCES

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39 (1):3–20, 2020.

[2] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[4] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[5] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[6] Christian Bizer and Andreas Schultz. The r2r framework: Publishing and discovering mappings on the web. *COLD*, 665:97–108, 2010.

[7] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[8] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[10] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[11] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[12] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts, 2021. URL https://arxiv.org/abs/2102.08981.

[13] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[14] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026. URL https://arxiv.org/abs/2510.25889.

[15] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions, 2023. URL https://arxiv.org/abs/2311.12793.

[16] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

[17] Shuo Cheng, Liqian Ma, Zhenyang Chen, Ajay Mandlekar, Caelan Garrett, and Danfei Xu. Generalizable domain adaptation for sim-and-real policy co-training. *arXiv preprint arXiv:2509.18631*, 2025.

[18] Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

[19] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

[20] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

[21] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[22] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021. URL https://arxiv.org/abs/2109.13396.

[23] Kaipeng Fang, Weiqing Liang, Yuyang Li, Ji Zhang, Pengpeng Zeng, Lianli Gao, Jingkuan Song, and

Heng Tao Shen. Sim-and-human co-training for data-efficient and generalizable robotic manipulation, 2026. URL https://arxiv.org/abs/2601.19406.

[24] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

[25] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1634–1643, 2021.

[26] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13137–13146, 2020.

[27] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1643–1653, 2021.

[28] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[29] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

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

[31] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

[32] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018. URL https://arxiv.org/abs/1806.10293.

[33] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[34] 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. *arXiv preprint arXiv:2403.12945*, 2024.

[35] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[36] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[37] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*, 2020.

[38] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[39] Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. *arXiv preprint arXiv:2411.11839*, 2024.

[40] Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[41] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[42] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[43] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025.

[44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[45] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969. IEEE, 2024.

[46] Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, et al. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *arXiv preprint*

arXiv:2503.24361, 2025.

[47] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[48] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[49] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[50] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[51] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[52] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

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

[54] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[55] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[56] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[57] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[58] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[59] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025.

[60] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[61] Gemini Robotics Team, Krzysztof Choromanski, Coline Devin, Yilun Du, Debidatta Dwibedi, Ruiqi Gao, Abhishek Jindal, Thomas Kipf, Sean Kirmani, Isabel Leal, et al. Evaluating gemini robotics policies in a veo world simulator. *arXiv preprint arXiv:2512.10675*, 2025.

[62] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[63] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[64] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020.

[65] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[66] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[67] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation, 2024. URL https://arxiv.org/abs/2403.03949.

[68] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al.

Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[70] Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels. *arXiv preprint arXiv:2503.22634*, 2025.

[71] Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, Fan Xu, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 192–198. IEEE, 2025.

[72] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[73] Jonathan Yang, Chelsea Finn, and Dorsa Sadigh. Invariance co-training for robot visual generalization. *arXiv preprint arXiv:2512.05230*, 2025.

[74] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. Inerf: Inverting neural radiance fields for pose estimation, 2021. URL https://arxiv.org/abs/2012.05877.

[75] Albert Yu, Adeline Foote, Raymond Mooney, and Roberto Martín-Martín. Natural language can help bridge the sim2real gap. *arXiv preprint arXiv:2405.10020*, 2024.

[76] Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025.

[77] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021. URL https://arxiv.org/abs/1910.10897.

[78] Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. Rlinf-vla: A unified and efficient framework for reinforcement learning of vision-language-action models, 2026. URL https://arxiv.org/abs/2510.06710.

[79] Kaifeng Zhang, Shuo Sha, Hanxiao Jiang, Matthew Loper, Hyunjong Song, Guangyan Cai, Zhuo Xu, Xiaochen Hu, Changxi Zheng, and Yunzhu Li. Real-to-sim robot policy evaluation with gaussian splatting simulation of soft-body interactions. *arXiv preprint arXiv:2511.04665*, 2025.

[80] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks, 2024. URL https://arxiv.org/abs/2412.18194.

[81] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

[82] Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. *arXiv preprint arXiv:2509.25756*, 2025.

[83] Siyuan Zhou, Yilun Du, Yuncong Yang, Lei Han, Peihao Chen, Dit-Yan Yeung, and Chuang Gan. Learning 3d persistent embodied world models. *arXiv preprint arXiv:2505.05495*, 2025.

[84] Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: Learning interactive real-robot action simulators. *arXiv preprint arXiv:2406.14540*, 2024.

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

VII. Appendix

A. Real-world Environment Setup

Fig. 8 illustrates our real-world evaluation setup. The system consists of a table-top workspace, a Franka Emika Panda robot mounted on the table, and a fixed RGB camera. We use the RGB channels captured by the camera as the visual input to the VLA model.

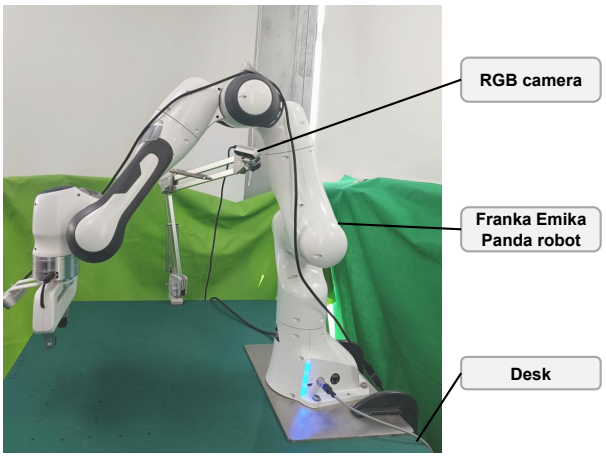

Fig. 8: **Real-world Setup.** The real-world evaluation platform includes a tabletop workspace, a Franka Panda robotic manipulator fixed to the table, and a RGB camera for visual perception. All objects are positioned on the table surface.

The robot is equipped with seven actuated joints and a parallel-jaw gripper with open/close capability. Control is performed in an end-effector delta-pose space: at each timestep, we command a relative end-effector pose with respect to the current pose, and compute the corresponding joint updates using inverse kinematics (IK). The translational components are specified in the robot base frame, while the rotational components are represented as roll–pitch–yaw (RPY) angles relative to the current end-effector orientation. Gripper actuation is controlled separately using a binary open/close signal. Overall, the action space is 7-dimensional.

B. Evaluation Details

1) Visualization of All Tasks: Fig. 9 visualizes the four table-top manipulation tasks evaluated in our experiments. `Pick and Place` requires the robot to grasp an object from the table and place it into a bowl. `Push Cube` involves pushing one cube out of three candidates with different colors according to a language instruction. `Open Drawer` requires opening a closed drawer placed on the table, while `Close Drawer` requires closing an initially opened drawer.

2) Manipulated Objects: Fig. 10 shows the manipulated objects used in both simulation and real-world experiments. The detailed settings are summarized below:

- `Pick and Place`: In simulation, we use the same set of 25 objects as in the environment proposed by Liu et al. [43]. In the real world, objects are divided into two categories: regular-shaped and irregular-shaped. Regular-shaped objects consist of toy fruits and vegetables, while irregular-shaped objects include bowls and gloves. Notably, irregular-shaped objects are not included in the real-world expert demonstrations. For in-distribution evaluation, we select four regular-shaped objects for testing.
- `Push Cube`: In simulation, we train on five colored cubes, as shown in Fig. 10. In the real-world setup, we also use five colors. However, expert demonstrations are collected only for three colors (purple, yellow, and pink), while orange and green cubes are excluded from the demonstration data. During evaluation, three colors are randomly selected from the five available colors.
- `Open/Close Drawer`: In the real world, we use the drawer shown in Fig. 10. In simulation, we construct a corresponding URDF model with matched geometric proportions.

3) Objects Initial States: Fig. 11 illustrates the randomized regions for the four real-world tasks. The detailed configurations are as follows:

- `Pick and Place`: The bowl is randomly placed within a 10×20 cm rectangular region, indicated by the orange area in Fig. 11. For each episode, one object is selected from a predefined object set, and its center is randomly placed within a 20×25 cm rectangular region, indicated by the blue area in Fig. 11. To facilitate controlled evaluation, both the bowl and object regions are discretized into grids with a minimum resolution of 5 cm. All objects are placed on grid points, and the same set of initial configurations is used across different methods.
- `Push Cube`: For each evaluation episode, three cubes are randomly selected from all available colors and randomly ordered. The cubes are initially placed with a spacing of 15 cm, followed by a random perturbation within a 5×5 cm region, as indicated by the orange area in Fig. 11. The language instruction specifies one of the three colors. All experiments follow the same color permutations and spatial configurations.
- `Open Drawer`: The front edge of the closed drawer is placed within the orange region shown in Fig. 11, with the drawer orientation initially aligned parallel to the short edge of the table. A random rotational perturbation of up to $15°$ is then applied. We uniformly sample 10 predefined initial configurations, which are shared across all evaluations.
- `Close Drawer`: Similar to `Open Drawer`, the drawer is initially opened by approximately 10 cm, and its front edge is placed within the same orange region with up to $15°$ of rotational perturbation. The same set of 10 predefined configurations is used for evaluation to ensure fair comparison.

4) Robot Initial State: Unless otherwise specified, the Franka Emika Panda robot is initialized in a fixed default configuration across all experiments, as shown in Fig. 8. Here

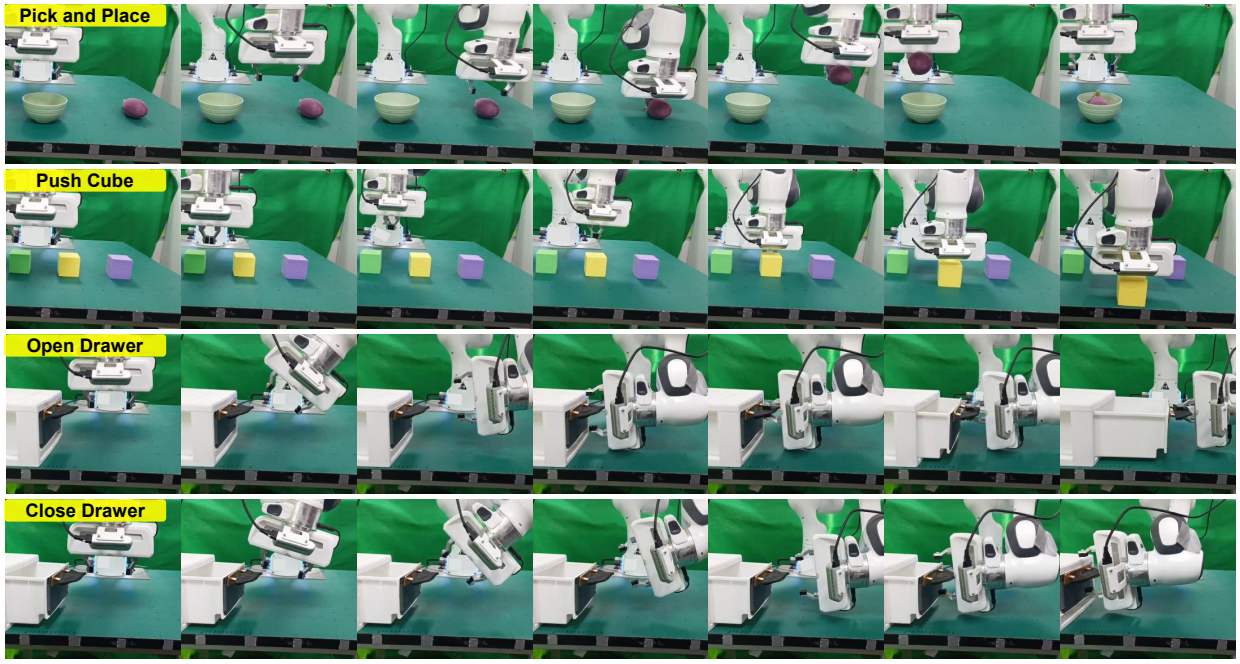

Fig. 9: **Visualization of Four Tabletop Manipulation Tasks.** For each task, we present one successful trajectory and uniformly sample seven frames along the execution. Each row corresponds to a single trajectory shown from start to completion.

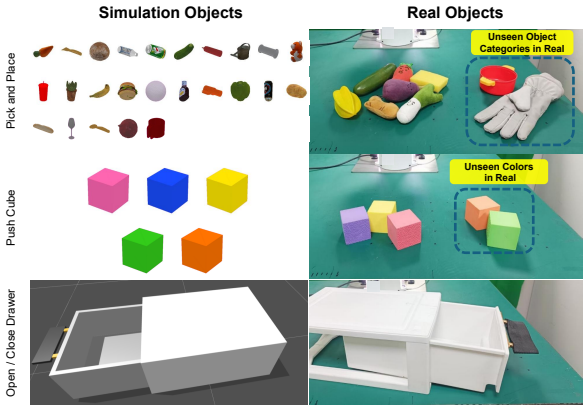

Fig. 10: **Manipulated Objects in Simulation and the Real World.** The left panel shows the objects used in simulation, while the right panel presents the real-world objects. All simulated objects are used during training. The real-world objects are divided into training objects and unseen objects for generalization evaluation.

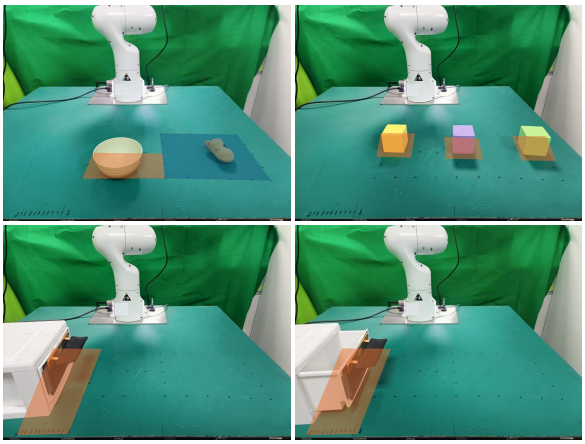

Fig. 11: **Initial Regions for Manipulative Objects.** For the `Pick and Place` task, the bowl is placed within the orange region, while the objects are initialized in the blue region. For the `Push Cube` task, each cube is initialized within its corresponding orange region. For the `Open / Close Drawer` tasks, the front edge of the drawer is initialized within the orange region.

we describe additional robot initial states used in the generalization experiments. Specifically, we focus on the `Pick and Place` task and select four representative objects, each with a fixed object initialization. For each object, we perturb the robot tool center point (TCP) by applying a rotation of $\pm 30^\circ$ around the vertical axis, together with a translational offset of 5 cm along a single Cartesian direction. The perturbations include forward, backward, leftward, rightward, and upward translations, resulting in five distinct perturbed initial states. Each perturbation combines one directional translation with the corresponding rotational offset. These perturbations are summarized in Table III. All other aspects of the environment and policy remain unchanged.

C. Simulation Training

1) Reward Function Design: We detail the reward function design for each simulation task in this section.

- `Pick and Place`: This task is decomposed into two sequential stages: *grasping* and *placing*, which

Perturbation ID	Translation (cm)	Rotation (deg)
P1	$(+5,\ 0,\ 0)$ (forward)	$+30°$
P2	$(-5,\ 0,\ 0)$ (backward)	$-30°$
P3	$(0,\ +5,\ 0)$ (left)	$+30°$
P4	$(0,\ -5,\ 0)$ (right)	$-30°$
P5	$(0,\ 0,\ +5)$ (upward)	$+30°$

TABLE III: **Robot initial state perturbations applied to the TCP in the `Pick and Place` task.** Translations are defined in the world frame, and rotations are applied around the vertical (z) axis.

are indicated by the binary states `is_grasped` and `is_placed`, respectively. We design two types of reward functions: a dense reward and a sparse reward. The dense reward is defined as

$$\mathcal{R} = \min\Big\{\mathbb{I}_{\text{success}},\ \big(\mathbb{I}_{\text{grasped}}(1+\mathcal{R}_d(d_2)) + \mathcal{R}_d(d_2)\big)\Big\}, \tag{9}$$

where $\mathbb{I}_{\text{grasped}}$ and $\mathbb{I}_{\text{success}}$ are indicator functions denoting whether the object has been successfully grasped and placed, respectively. The shaping term $\mathcal{R}_d(x) = 1 - \tanh(10x)$ provides a smooth distance-based reward that asymptotically approaches 1 as the distance decreases. Here, d_1 and d_2 denote the distance between the gripper and the object, and the distance between the object and the target container, respectively. For the sparse reward, we assign a reward of 0.2 at the moment when grasping succeeds, and a reward of 1 upon successful placement. If the object leaves the target container after a successful placement due to external disturbances, a penalty of -0.4 is applied. All other timesteps receive zero reward. We use dense reward when training OpenVLA and use sparse reward when training $\pi_{0.5}$.

- `Push Cube`: The objective of this task is to push a designated target cube into a predefined goal region on the table. The dense reward consists of three components. First, a *reaching reward* encourages the Tool Center Point (TCP) to approach a pre-defined pushing pose behind the cube along the pushing direction:

$$r_{\text{reach}} = 1 - \tanh(5 \cdot \|\mathbf{p}_{\text{tcp}} - \mathbf{p}_{\text{push}}\|_2)\,, \tag{10}$$

where $\mathbf{p}_{\text{push}}$ is defined as a point offset from the cube center by one half cube length plus a small margin along the pushing axis. Second, once the TCP is sufficiently close to the pushing pose, a *placement reward* is activated to encourage the cube to move toward the goal region:

$$r_{\text{place}} = 1 - \tanh\Big(5 \cdot \|\mathbf{p}^{xy}_{\text{cube}} - \mathbf{p}^{xy}_{\text{goal}}\|_2\Big)\,, \tag{11}$$

where only planar (x, y) distances are considered. This term is gated by a proximity condition to ensure that the agent first establishes contact before being rewarded for object motion. Finally, a sparse *success bonus* is assigned once the cube is pushed beyond the goal center along the pushing direction and remains within a tolerance band orthogonal to it:

$$r = \begin{cases} 3.0, & \text{if success,} \\ r_{\text{reach}} + r_{\text{place}}, & \text{otherwise.} \end{cases} \tag{12}$$

- `Open Drawer`: In this task, the reward is defined over three stages corresponding to reaching, opening progress, and task completion. First, a reaching reward encourages the TCP to approach the drawer handle:

$$r_{\text{reach}} = 1 - \tanh(5 \cdot \|\mathbf{p}_{\text{tcp}} - \mathbf{p}_{\text{handle}}\|_2)\,. \tag{13}$$

Second, an *opening reward* is defined based on the normalized drawer joint position (open fraction):

$$r_{\text{open}} = 2 \cdot \text{open_frac}, \tag{14}$$

where the open fraction is computed by linearly normalizing the drawer joint position between its minimum and maximum limits. Once the drawer begins to open, the reaching reward is saturated to a constant value to avoid conflicting gradients. The total dense reward is given by:

$$r = r_{\text{reach}} + r_{\text{open}}. \tag{15}$$

A terminal success reward of 5.0 is assigned when the drawer is opened beyond a high open-fraction threshold (e.g., 90% of its range).

- `Close Drawer`: The Close Drawer task is initialized from an open state and rewards progress toward closing the drawer. Instead of absolute position, we define a *progress-based reward* using the change in open fraction between consecutive time steps:

$$\Delta = \text{open_frac}_{t-1} - \text{open_frac}_t. \tag{16}$$

The dense reward is then computed as:

$$r = \alpha \cdot \text{clip}(\Delta, -1, 1) - \beta, \tag{17}$$

where α is a scaling factor for closing progress and β is a small time penalty that encourages faster completion. Once the drawer is closed below a predefined threshold on the open fraction, a terminal success reward of 5.0 is issued and overrides the dense shaping terms.

For all tasks, dense rewards are normalized by their respective maximum achievable reward values to ensure comparable reward scales across tasks during multi-task training.

2) Performance in Simulation during RL Training: Fig. 12 shows the success rates of different models on each task in the simulation environment during RL training. After RL fine-tuning, all VLA models achieve substantial performance improvements in simulation compared to their pre-RL counterparts.

D. Implementation Details

For OpenVLA, we fine-tune the model using LoRA with a rank of 32, whereas $\pi_{0.5}$ is fine-tuned with full-parameter updates. All models are optimized using the AdamW optimizer [44]. The detailed hyperparameters are summarized in Table IV.

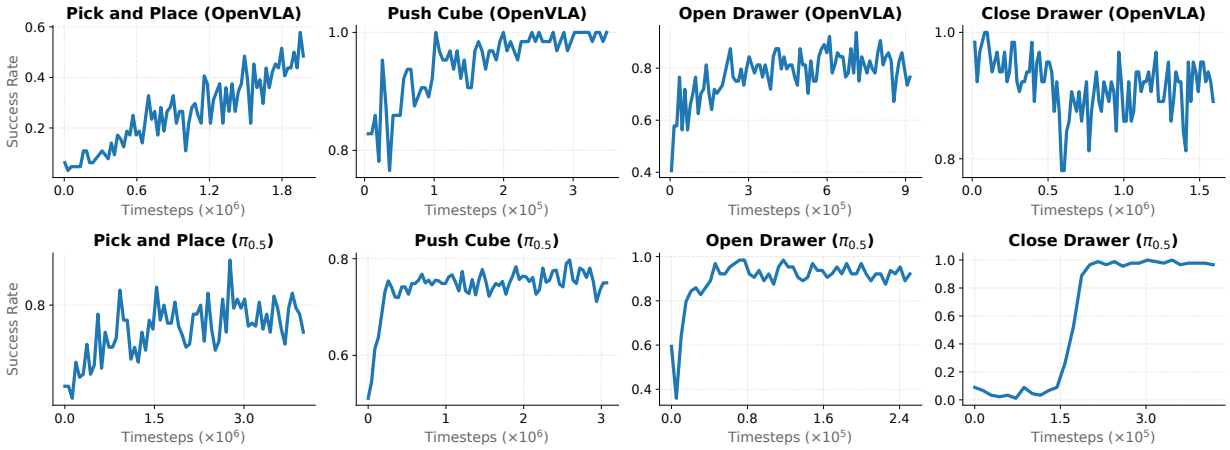

Fig. 12: **Simulation Training Results.** We report the simulation success rates across all settings during RL training.

Parameter Names	Setting	Pick and Place	Push Cube	Open Drawer	Close Drawer
General	Number of Real Demos	50	50	20	30
OpenVLA	Co-training ratio α	0.5	0.5	0.5	0.5
	SFT learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
	Regularization weight β	0.1	0.01	0.01	0.01
	Actor learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}
	Critic learning rate	3×10^{-3}	3×10^{-3}	3×10^{-3}	3×10^{-3}
$\pi_{0.5}$	Co-Training Ratio α	0.5	0.95	0.95	0.98
	SFT learning rate	2.5×10^5	2.5×10^5	2.5×10^5	2.5×10^5
	SFT lr schedule	Cosine Decay	Cosine Decay	Cosine Decay	Cosine Decay
	Regularization Weight β	1.0	0.2	1.0	0.1
	Actor learning rate	4×10^{-6}	10^{-6}	4×10^{-6}	7.91×10^{-6}
	Critic learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	1.55×10^{-4}

TABLE IV: **Hyperparameter settings for different VLA models and tasks.** We report task-specific hyperparameters used for OpenVLA and $\pi_{0.5}$ across four real-world manipulation tasks.