# REROUTING THE FLOW: VERSATILE TEST-TIME VLA ADAPTATION VIA SOURCE DISTRIBUTION LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision-language-action (VLA) agents acquire broad manipulation capabilities from large-scale robot data, yet remain brittle under distribution shifts at deployment. Our diagnostic analyses show that failed rollouts can retain task-directed behavior and that denoising perturbations occasionally yield success, suggesting that some failures reflect unreliable access to pretrained capabilities rather than their absence. Motivated by this insight, we propose REFLOWTTA, a versatile test-time adaptation framework that learns the source distribution of initial noise of a flow-based VLA, shifting probability mass toward states associated with better execution outcomes. Task feedback and optional human corrections guide this shift, allowing accumulated experience to shape subsequent sampling and make successful behaviors more accessible. This turns adaptation across objects, scenes, and robot embodiments into a common problem of learning where generation should begin, without relearning how actions are produced. Across three simulation benchmarks and two real-world robot platforms, REFLOWTTA consistently outperforms representative test-time adaptation methods with limited deployment interactions. Further analyses show that adaptation stabilizes and that the gains depend on the direction of the learned source shift instead of on generic perturbations. We release the core implementation at an anonymous repository.

## 1 INTRODUCTION

Vision-Language-Action (VLA) agents acquire broad manipulation capabilities through large-scale robot learning (Black et al., 2025b; Liu et al., 2025b). However, they often struggle with unfamiliar objects, scenes, and robot embodiments during deployment (Fei et al., 2026; Kwok et al., 2025). Improving reliability under these out-of-distribution (OOD) conditions therefore remains a central requirement for general-purpose embodied agents (Ghosh et al., 2024; O'Neill et al., 2024).

In this work, we seek to improve OOD task execution without costly data collection and VLA retraining. This is motivated by two observations: ❶ *Failed rollouts retain task-relevant behavior.* In our OOD studies (*Appendix* B.1), though fails to grasp, the agent approaches the correct object and continues along the intended grasp-and-transfer sequence. This preserved task behavior prompts us to explore alternative denoising trajectories, which raises the second observation: ❷ *Perturbing the denoising process occasionally produces successful rollouts.* We inject Gaussian noise to initial states and early denoising steps, yielding 4/50 and 3/50 successful rollouts on two OOD tasks (*Appendix* B.2).

These studies suggest an insight that pretrained VLAs actually hold the capabilities to handle some OOD tasks, yet struggle to translate them into successful execution (Fig. 1). Specifically, for a diffusion- or flow-based VLA agent (Liu et al., 2025b; Black et al., 2025b), the resulting action distribution depends both on a source distribution which determines how initial noise states are sampled, and a learned noise-to-action mapping to transform these states into actions (Chi et al., 2023; Lipman et al., 2023; Black et al., 2025a). Thus, a frozen VLA agent (*i.e.*, fixed noise-to-action mapping) does not imply a determined output. The adaptation can be formulated as learning a source distribution that turns **unguided perturbations with only occasional success** into **targeted sampling of favorable initial states** that could unlock the full potential of a pretrained VLA agent.

From this perspective, we propose REFLOWTTA, a test-time adaptation approach that reroutes flow-based action generation by adapting the initial-noise distribution. **First**, we define a Gaussian distribution over coordinates in a fixed low-dimensional subspace of the full source space, with a learnable mean vector and fixed unit covariance. At each generation step, an independent sample from this



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

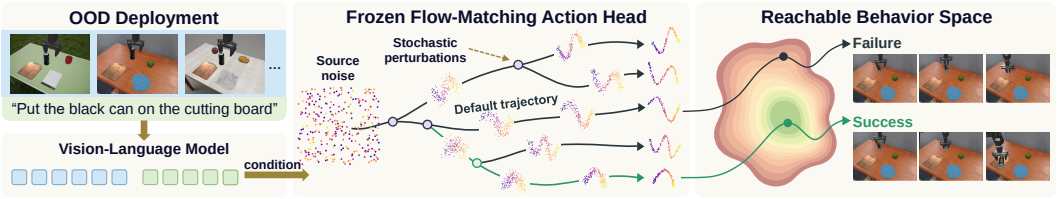



Figure 1: **Motivation.** Different denoising trajectories under the same frozen VLA can yield distinct execution outcomes, with successful behaviors remaining attainable even when the default trajectory fails.

Gaussian replaces only the corresponding coordinates of the original source noise. This allows us to shift the distribution of initial states by updating only the low-dimensional mean, while preserving the covariance of the full source distribution (§ 3.1). **Next**, we use the resulting noise to initialize the frozen flow and generate an action chunk. After each chunk-level execution, the task reward is paired with the low-dimensional coordinates used to generate that chunk, with clipped PPO (Schulman et al., 2017) updating the mean to favor coordinates associated with better task outcomes. Note that the updated mean is retained across steps and episodes under the same deployment condition, allowing past feedback to guide subsequent action generation (§ 3.2). **Finally**, we introduce optional human corrections which directly specify desired actions to complement execution-based rewards. Since human corrections are expressed in action space, we integrate the frozen flow in reverse to recover their source noise states, which are then projected onto the same low-dimensional subspace. The resulting coordinates provide additional positive targets in source space to update the same mean vector by maximizing the likelihood (§ 3.3). This turns corrective actions into guidance on where subsequent flow trajectories should begin, reducing reliance on agent-generated trial and error.

Overall, REFLOWTTA turns the low-dimensional source distribution into a deployment-time adaptation interface for frozen VLAs. This offers several advantages: **i)** *Versatility*: unlike methods tailored to specific variations (Yoneda et al., 2022; Dey et al., 2025; Zhu et al., 2023), REFLOWTTA acts on the shared flow-based action-generation process, which formulates adaptation to diverse OOD conditions across objects, scenes, and robot embodiments, as a common distribution learning problem. **ii)** *Persistence*, instead of re-selecting among action candidates during each decision step (Kwok et al., 2025; Song et al., 2026), REFLOWTTA retains the mean vector across episodes, allowing past feedback to consistently improve subsequent action generation. **iii)** *Lightweight*, both task rewards and human corrections update only the low-dimensional source mean vector, avoiding direct VLA parameter updates which can become unstable due to scarce deployment data (Liu et al., 2026); **iv)** *Flexibility*, human corrections are optional rather than a prerequisite for adaptation, while prior work requires pre-collected demonstrations or successful trajectories (Sridhar et al., 2025).

REFLOWTTA is evaluated on ManiSkill3 (Tao et al., 2025), LIBERO (Liu et al., 2023), and LIBERO-Plus (Fei et al., 2026), together with two real-world robot platforms, covering object- and scene-level shifts, cross-embodiment transfer, and compound OOD conditions. It outperforms representative TTA methods, improving Final SR over the SOTA by **11.8** points on ManiSkill3, **14.1** points on LIBERO, **7.9** points on LIBERO-Plus, and up to **17.8** points in real-world evaluation. Further analyses show that shifting the source distribution through low-dimensional mean updates improves both adaptation efficiency and stability. The code is available at an anonymous repository.

## 2 RELATED WORK

**Robot Policy under OOD Conditions.** Reinforcement learning improves pretrained robot policies through additional target-domain interaction (Yuan et al., 2025; Ren et al., 2025; Hu et al., 2024; Chen et al., 2025; Xu et al., 2025). However, the resulting policies remain fixed during deployment, and handling new OOD conditions requires further data collection and training (Hu et al., 2025a). Some approaches are designed for particular types of variation. For example, some methods specifically address visual variations such as changes in object appearance, lighting, and viewpoint (Yoneda et al., 2022; Dey et al., 2025; Zhu et al., 2023), while constraint-based methods incorporate geometric and physical requirements into action generation (Huang et al., 2024; Deng et al., 2025). Their applicability therefore depends on the variations targeted by each design. Another line trains context-identification modules in randomized simulations to infer environment properties from interaction history and adjust policy behavior accordingly (Yu et al., 2017; Kumar et al., 2021). Although such methods can adjust behavior online, the range of conditions they can accommodate is also specified during training. In contrast, REFLOWTTA requires no predefined variations and

can adapt directly from deployment feedback. It formulates unfamiliar objects, scenes, and embodiments as a common source-distribution learning problem, applying the same learning procedure across heterogeneous OOD conditions without introducing separate adaptation mechanisms.

**Test-Time Adaptation.** Test-time adaptation has been widely studied in computer vision and large language models (Sun et al., 2020; Hardt & Sun, 2024; Hu et al., 2025b). Existing approaches learn from unlabeled test inputs or model predictions through pseudo-labeling and self-supervision (Sun et al., 2020; Hu et al., 2025b; Goyal et al., 2022), entropy minimization (Wang et al., 2021; Niu et al., 2022), distribution alignment (You et al., 2025), or prediction consistency (Wang et al., 2022; Zhang et al., 2022). Applying these objectives to VLA agents is less straightforward. Consistent action predictions do not necessarily imply successful task execution, since a robot may repeatedly produce task-directed but unsuccessful behavior (Agia et al., 2024). For VLA agents, recent work explores verification-based action selection and value-guided sampling (Kwok et al., 2025; Song et al., 2026), as well as in-context adaptation using demonstration trajectories (Sridhar et al., 2025; Zhang et al., 2026). Another line learns from deployment feedback by directly updating VLA parameters (Liu et al., 2026) or optimizing perturbations of generated actions (Zang et al., 2026). REFLOWTTA instead adapts only a low-dimensional source mean, without requiring pre-collected demonstration context, full-model backpropagation through the VLA or repeated generation and selection of action candidates, enabling lightweight adaptation and simpler real-world deployment.

**Source Noise Selection and Optimization.** Diffusion and flow-matching models generate structured outputs by transforming samples drawn from a source distribution (Ho et al., 2020; Lipman et al., 2023). Even with fixed model parameters and conditioning inputs, different initial noise samples can lead to different generation trajectories and results (Song et al., 2021a;b). Studies in image generation show that this choice affects visual quality, semantic alignment, and structural fidelity (Guo et al., 2024; Zhou et al., 2025). Based on these observations, some methods sample and rank candidate noises to select favorable initializations (Samuel et al., 2024; Ma et al., 2025; Guo et al., 2026). Others directly optimize the initial noise according to generation objectives, improving image quality or condition alignment without updating the generative model (Eyring et al., 2024; Miao et al., 2025). In robot learning, diffusion- and flow-based VLAs also transform source noise into continuous action chunks conditioned on observations and language instructions (Black et al., 2025b; Liu et al., 2025b), typically sampling source noise from a fixed prior (Chi et al., 2023; Ghosh et al., 2024). More closely related to our work, some approaches condition source noise on task representations during policy training or learn a state-conditioned noise policy through RL post-training (Dong et al., 2025; Wagenmaker et al., 2025). REFLOWTTA instead updates only a shared, low-dimensional source mean from limited chunk-level deployment feedback, while preserving the source covariance. The resulting distribution is retained across steps and episodes under the same OOD condition, allowing experience from earlier executions to shape later sampling.

## 3 METHOD

**Problem Formulation.** Consider a pretrained VLA agent $\pi_\theta$ with parameters $\theta$. At each decision step $t$, the agent receives a natural language instruction $\ell$, its proprioceptive state $\boldsymbol{q}_t$, and a multi-view visual observation $\boldsymbol{O}_t = \{\boldsymbol{I}_t^n\}_{n=1}^N$, where $\boldsymbol{I}_t^n \in \mathbb{R}^{3\times H\times W}$ denotes the RGB image from the $n$-th of $N$ camera views. By default, the flow-matching action head transforms standard Gaussian source noise $\boldsymbol{\epsilon}_t \in \mathbb{R}^{T\times D}$, with entries independently sampled from $\mathcal{N}(0,1)$, into a continuous action chunk conditioned on the inputs (Black et al., 2025b;a):

$$\boldsymbol{A}_t = \pi_\theta(\boldsymbol{O}_t, \boldsymbol{q}_t, \ell; \boldsymbol{\epsilon}_t) = [\boldsymbol{a}_{t,1}, \ldots, \boldsymbol{a}_{t,T}] \in \mathbb{R}^{T\times D}, \tag{1}$$

where $\boldsymbol{a}_{t,j} \in \mathbb{R}^D$ is the $j$-th action predicted at decision step $t$, $T$ is the action horizon, and $D$ is the action dimension. The agent executes the first $T_e \le T$ actions before receiving chunk-level task feedback $r_t$ and acquiring a new observation. Our goal is to improve OOD task execution by learning the source-noise distribution from limited deployment feedback, while keeping $\theta$ fixed.

**Overview (Fig. 2).** REFLOWTTA reroutes action generation by adapting the source distribution of the frozen flow-matching action head, with its mean restricted to a fixed low-dimensional subspace (§ 3.1). Sampled noise is transformed into action chunks, whose execution rewards update the mean through clipped PPO (Schulman et al., 2017) to guide subsequent sampling (§ 3.2). To complement these reward-based updates, available human corrections are mapped back through inverse flow and projection, providing positive targets to update the same mean via likelihood maximization (§ 3.3).

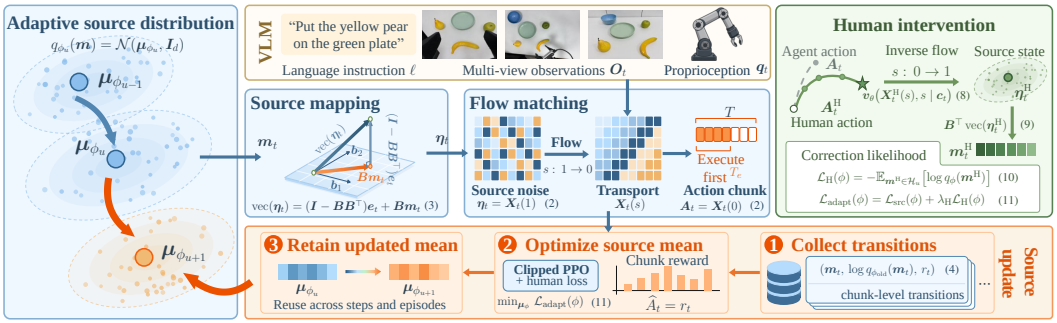



Figure 2: **Overview.** REFLOWTTA initializes a frozen action head using a Gaussian distribution whose mean is adapted within a fixed low-dimensional subspace (§ 3.1). Task rewards from chunk-level execution update the mean through clipped PPO (§ 3.2), while optional human corrections are mapped back through inverse flow to provide additional source-space targets for updating the same mean via likelihood maximization (§ 3.3). The mean vector is retained across steps and episodes under the same OOD condition, allowing accumulated feedback to guide subsequent action generation. The pretrained VLA remains unchanged throughout adaptation.

### 3.1 LOW-DIMENSIONAL SOURCE DISTRIBUTION FOR FLOW REROUTING

At each decision step $t$, the frozen flow-matching action head transports a source state $\boldsymbol{\eta}_t \in \mathbb{R}^{T\times D}$ to an action chunk by integrating the conditional velocity field $\boldsymbol{v}_\theta$ (Black et al., 2025b):

$$\frac{\mathrm{d}\boldsymbol{X}_t(s)}{\mathrm{d}s} = \boldsymbol{v}_\theta\big(\boldsymbol{X}_t(s), s \mid \boldsymbol{c}_t\big), \qquad \boldsymbol{X}_t(1) = \boldsymbol{\eta}_t, \qquad \boldsymbol{X}_t(0) = \boldsymbol{A}_t, \tag{2}$$

where $s\in[0,1]$ denotes flow time from source to action $(1\to 0)$, $\boldsymbol{X}_t(s)\in\mathbb{R}^{T\times D}$ is the intermediate action state, and $\boldsymbol{c}_t = (\boldsymbol{O}_t, \boldsymbol{q}_t, \ell)$ collects the conditioning inputs. For fixed $\theta$ and $\boldsymbol{c}_t$, the source state determines the flow trajectory and resulting action chunk. REFLOWTTA adapts action generation by learning the distribution over these initial states, while keeping the velocity field unchanged.

To learn this distribution from limited deployment feedback, we restrict mean adaptation to a low-dimensional subspace spanned by the orthonormal columns of $\boldsymbol{B}\in\mathbb{R}^{TD\times d}$, where $d \ll TD$ denotes the subspace dimension. The subspace coordinates follow $q_\phi(\boldsymbol{m}) = \mathcal{N}(\boldsymbol{\mu}_\phi, \boldsymbol{I}_d)$, with $\boldsymbol{\mu}_\phi \in \mathbb{R}^d$ being the only parameter updated during deployment. At each decision step, we independently sample $\boldsymbol{m}_t \sim q_\phi(\boldsymbol{m})$ and Gaussian noise $\boldsymbol{\epsilon}_t$. The shifted initial noise state $\boldsymbol{\eta}_t$ is then constructed by replacing the subspace coordinates of $\boldsymbol{\epsilon}_t$ with $\boldsymbol{m}_t$ while retaining its orthogonal component:

$$\boldsymbol{\eta}_t = \mathrm{vec}^{-1}[(\boldsymbol{I}_{TD} - \boldsymbol{B}\boldsymbol{B}^\top)\boldsymbol{e}_t + \boldsymbol{B}\boldsymbol{m}_t], \qquad \boldsymbol{e}_t = \mathrm{vec}(\boldsymbol{\epsilon}_t), \tag{3}$$

where $\mathrm{vec} : \mathbb{R}^{T\times D} \to \mathbb{R}^{TD}$ is a fixed vectorization operator, and $\mathrm{vec}^{-1}$ is its inverse. This yields $\mathrm{vec}(\boldsymbol{\eta}_t) \sim \mathcal{N}(\boldsymbol{B}\boldsymbol{\mu}_\phi, \boldsymbol{I}_{TD})$, which preserves the original covariance while confining the learnable mean to the subspace spanned by $\boldsymbol{B}$. Initializing $\boldsymbol{\mu}_{\phi_0} = \boldsymbol{0}_d$ recovers the original source distribution. Task feedback then guides $\boldsymbol{\mu}_\phi$ to favor initial states associated with better execution outcomes.

### 3.2 DEPLOYMENT-TIME SOURCE DISTRIBUTION LEARNING

REFLOWTTA learns the source distribution from chunk-level task feedback by treating $q_\phi(\boldsymbol{m})$ as a deployment-time policy over source coordinates, and uses a modified PPO objective to update its mean $\boldsymbol{\mu}_\phi$, so as to shift the source coordinates towards regions associated with better task outcomes.

For rollout collection, let $q_{\phi_{\mathrm{old}}}$ denote a fixed copy of the current source distribution, which also serves as the reference during subsequent optimization. At decision step $t$, a source coordinate $\boldsymbol{m}_t \sim q_{\phi_{\mathrm{old}}}(\boldsymbol{m})$ is sampled and used to construct $\boldsymbol{\eta}_t$ according to Eq. 3. The frozen VLA then generates an action chunk $\boldsymbol{A}_t$, of which the first $T_e \le T$ actions are executed before receiving task reward $r_t$ and acquiring new observations. We record this interaction as a chunk-level rollout sample $\mathcal{T}_t$:

$$\mathcal{T}_t = (\boldsymbol{m}_t,\ \log q_{\phi_{\mathrm{old}}}(\boldsymbol{m}_t),\ r_t), \tag{4}$$

where $\log q_{\phi_{\mathrm{old}}}(\boldsymbol{m}_t)$ is the stored log-probability of the sampled source coordinates under the behavior distribution. Each sample pairs the coordinates $\boldsymbol{m}_t$ with the reward $r_t$ obtained by executing the first $T_e$ actions of the corresponding chunk.

To optimize the mean without estimating a critic from limited deployment data, we adopt a value-free actor update which uses the immediate chunk reward as the advantage estimate $\widehat{A}_t = r_t$, with discount factor $\gamma = 0$ (Suhr & Artzi, 2023). Specifically, for each collected rollout sample, the joint log-probability under the current source distribution is formulated as:

$$\log q_\phi(\boldsymbol{m}_t) = -\frac{1}{2}\|\boldsymbol{m}_t - \boldsymbol{\mu}_\phi\|_2^2 - \frac{d}{2}\log(2\pi). \tag{5}$$

Comparing this distribution with the stored behavior log-probability gives the likelihood ratio:

$$\rho_t(\phi) = \exp\big[\log q_\phi(\boldsymbol{m}_t) - \log q_{\phi_{\text{old}}}(\boldsymbol{m}_t)\big]. \tag{6}$$

This chunk-aligned objective provides direct credit assignment in source space. The source-distribution mean is optimized by minimizing the clipped PPO loss (Schulman et al., 2017):

$$\mathcal{L}_{\text{src}}(\phi) = -\mathbb{E}_t\Big[\min\big(\rho_t(\phi)\widehat{A}_t,\ \text{clip}\big(\rho_t(\phi), 1-\varepsilon_{\text{clip}}, 1+\varepsilon_{\text{clip}}\big)\widehat{A}_t\big)\Big], \tag{7}$$

where $\varepsilon_{\text{clip}}$ specifies the clipping range. The gradient updates only $\boldsymbol{\mu}_\phi$, leaving the source covariance and pretrained VLA unchanged. For each update round $u$ without human corrections, REFLOWTTA groups the next $K_{\text{roll}}$ chunk-level transitions into a rollout set $\mathcal{D}_u$. Each transition pairs a sampled source coordinate with the reward obtained after executing the first $T_e$ actions of the generated chunk. We then update the current mean $\boldsymbol{\mu}_\phi$ by minimizing Eq. 7 on $\mathcal{D}_u$ and use the updated distribution to collect the next batch. The mean is retained across episodes under the same OOD condition, where feedback from earlier interactions continues to guide subsequent source sampling.

### 3.3 SOURCE DISTRIBUTION LEARNING FROM HUMAN INTERVENTION

During real-world deployment, human intervention can provide corrective action chunks when the agent encounters failures, reducing the trials required for adaptation. Unlike agent-generated action chunks, human corrections are provided in action space instead of being sampled from the source distribution. The corresponding source coordinates and behavior probabilities are unavailable for PPO updates. To tackle this, REFLOWTTA inverts the flow dynamics to recover the low-dimensional source coordinate and uses it as supervised targets for learning the source distribution.

Let $\boldsymbol{A}_t^{\text{H}} \in \mathbb{R}^{T\times D}$ denote a corrective action chunk under conditioning $\boldsymbol{c}_t$. To recover its source state, we reverse the action-generation process in Eq. 2. Specifically, we keep $\boldsymbol{c}_t$ fixed and integrate the frozen velocity field from the action endpoint at $s=0$ to the source point at $s=1$(Grathwohl et al., 2019):

$$\frac{\mathrm{d}\boldsymbol{X}_t^{\text{H}}(s)}{\mathrm{d}s} = \boldsymbol{v}_\theta\big(\boldsymbol{X}_t^{\text{H}}(s), s \mid \boldsymbol{c}_t\big), \qquad \boldsymbol{X}_t^{\text{H}}(0) = \boldsymbol{A}_t^{\text{H}}, \qquad \boldsymbol{\eta}_t^{\text{H}} = \boldsymbol{X}_t^{\text{H}}(1). \tag{8}$$

The endpoint $\boldsymbol{\eta}_t^{\text{H}}$ is the recovered full-dimensional source state under the same conditioning and flow dynamics used for action generation. Since mean adaptation is restricted to the subspace spanned by $\boldsymbol{B}$, we project this state onto its orthonormal basis to obtain the correction target:

$$\boldsymbol{m}_t^{\text{H}} = \boldsymbol{B}^\top \text{vec}\big(\boldsymbol{\eta}_t^{\text{H}}\big). \tag{9}$$

Recovered intervention coordinates serve as positive targets in source space and are collected into an intervention set $\mathcal{H}_u$ for the $u$-th update. REFLOWTTA increases their likelihood under the source-space policy using the negative log-likelihood objective:

$$\mathcal{L}_{\text{H}}(\phi) = -\mathbb{E}_{\boldsymbol{m}^{\text{H}}\in\mathcal{H}_u}\big[\log q_\phi\big(\boldsymbol{m}^{\text{H}}\big)\big]. \tag{10}$$

With fixed unit covariance, minimizing this loss is equivalent to minimizing the average squared distance between $\boldsymbol{\mu}_\phi$ and the correction targets. Human corrections thus guide the source mean toward coordinates associated with corrective actions.

We combine this supervision with the reward-based objective from § 3.2:

$$\mathcal{L}_{\text{adapt}}(\phi) = \mathcal{L}_{\text{src}}(\phi) + \lambda_{\text{H}}\mathcal{L}_{\text{H}}(\phi), \tag{11}$$

where $\lambda_{\text{H}}$ controls the contribution of human corrections. Both terms update the same distribution mean $\boldsymbol{\mu}_\phi$. $\mathcal{L}_{\text{src}}$ uses execution rewards from autonomous samples, while $\mathcal{L}_{\text{H}}$ fits the coordinates from human corrections. When $\mathcal{H}_u$ is empty, we omit $\mathcal{L}_{\text{H}}$ and use only the autonomous PPO objective.

Table 2: **Adaptation on LIBERO.** SR@15 and Final SR are reported across four suites. Details in § 4.2.

| Agent | Spatial | | Object | | Goal | | Long | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SR@15 ↑ | Final SR ↑ | SR@15 ↑ | Final SR ↑ | SR@15 ↑ | Final SR ↑ | SR@15 ↑ | Final SR ↑ | SR@15 ↑ | Final SR ↑ |
| **Franka Panda** | | | | | | | | | | |
| Base VLA | – | 98.8 | – | 98.2 | – | 98.0 | – | 92.4 | – | 96.9 |
| **UR5e** | | | | | | | | | | |
| Base VLA | – | 35.4 | – | 30.6 | – | 12.0 | – | 3.0 | – | 20.3 |
| RoboMonkey | – | 46.4 | – | 41.2 | – | 12.6 | – | 3.4 | – | 25.9 |
| AAC | – | 43.6 | – | 37.8 | – | 12.4 | – | 3.6 | – | 24.4 |
| PDF | 45.7 | 50.2 | 39.6 | 44.0 | 11.3 | 10.8 | 2.9 | 2.6 | 24.9 | 26.9 |
| DSRL | 41.6 | 45.0 | 36.7 | 39.8 | 11.0 | 10.6 | 2.7 | 2.4 | 23.0 | 24.5 |
| **REFLOWTTA** | **61.1**$_{\pm 0.18}$ | **74.4**$_{\pm 0.14}$ | **54.0**$_{\pm 0.27}$ | **65.6**$_{\pm 0.20}$ | **15.8**$_{\pm 0.24}$ | **18.0**$_{\pm 0.28}$ | **4.8**$_{\pm 0.36}$ | **6.0**$_{\pm 0.24}$ | **33.9**$_{\pm 0.17}$ | **41.0**$_{\pm 0.13}$ |
| **KUKA IIWA** | | | | | | | | | | |
| Base VLA | – | 15.2 | – | 9.4 | – | 11.8 | – | 1.2 | – | 9.4 |
| RoboMonkey | – | 25.8 | – | 20.2 | – | 12.0 | – | 2.4 | – | 15.1 |
| AAC | – | 23.4 | – | 18.8 | – | 12.4 | – | 2.8 | – | 14.4 |
| PDF | 25.7 | 29.0 | 21.2 | 24.2 | 11.4 | 10.6 | 2.1 | 1.8 | 15.1 | 16.4 |
| DSRL | 22.1 | 24.6 | 17.8 | 19.6 | 11.8 | 11.2 | 2.5 | 2.2 | 13.6 | 14.4 |
| **REFLOWTTA** | **43.8**$_{\pm 0.20}$ | **54.8**$_{\pm 0.17}$ | **35.6**$_{\pm 0.23}$ | **44.6**$_{\pm 0.24}$ | **14.1**$_{\pm 0.31}$ | **15.8**$_{\pm 0.20}$ | **3.7**$_{\pm 0.27}$ | **4.6**$_{\pm 0.28}$ | **24.3**$_{\pm 0.18}$ | **30.0**$_{\pm 0.15}$ |

# 4 EXPERIMENT

## 4.1 EXPERIMENTAL SETUP

**Benchmarks and OOD Protocols.** REFLOWTTA is evaluated on three simulation benchmarks and real-world robotic platforms. To construct object- and scene-level OOD conditions in ManiSkill3 (Tao et al., 2025), we modify the widely used training protocol (Liu et al., 2025a) by holding out selected source and target object categories and table appearances during SFT. At test time, these held-out factors are combined with finer-grained variations in object positions and orientations and multiple distractors. Cross-embodiment adaptation is evaluated on the four standard 10-task suites of LIBERO (Liu et al., 2023) (Spatial, Object, Goal, and Long) by replacing the training-time Franka Panda with an unseen UR5e or KUKA IIWA. LIBERO-Plus (Fei et al., 2026) introduces controlled perturbations to the original LIBERO tasks and, following its standard evaluation setup, is evaluated using the same checkpoint as in our LIBERO experiment. These perturbations are further combined with an unseen UR5e or KUKA IIWA embodiment at test time, yielding compound OOD across task perturbations and embodiment shifts. Detailed configurations are provided in Appendices C.1–C.3.

**Baselines and Evaluation Metrics.** The comparison includes the $\pi_{0.5}$ (Black et al., 2025a) (our base agent) and four test-time methods covering two complementary axes: **i)** whether adaptation involves learning, where RoboMonkey (Kwok et al., 2025) and AAC (Liang et al., 2026b) keep the VLA agent frozen and instead improve execution through action-candidate selection and uncertainty-aware action chunking, respectively; **ii)** what is learned during adaptation: PDF (Zang et al., 2026) learns a perturbation head that adjusts action-token logits, DSRL (Wagenmaker et al., 2025) learns a state-conditioned source-noise policy through RL, and REFLOWTTA updates only a shared, low-dimensional source mean. We report Online **SR@K** as the average success rate of independent evaluations after each of the $K$ deployment episodes, capturing performance throughout adaptation rather than sporadic successes. It is reported only for methods that learn from deployment feedback across episodes. **Final SR** measures performance after adaptation using independent evaluation rollouts. Simulation results are averaged over five runs, with standard deviations reported.

## 4.2 SIMULATION RESULTS

**ManiSkill3: Object and Scene OOD.** Each task is adapted for 15 deployment episodes, with 50 rollouts sampled from the test-time distribution per evaluation. Table 1 shows that REFLOWTTA attains the highest Online SR@15 and Final SR under the ManiSkill3 OOD setting. It outperforms the strongest baseline, PDF (Zang et al., 2026), by **10.6%** in Online SR@15 (**49.2%** *vs* 38.6%) and **11.8%** in Final SR (**53.6%** *vs* 41.8%). Compared with the base VLA, REFLOWTTA further improves the Final SR by **19.8%**. These results highlight that adapting the low-dimensional source distribution can effectively exploit deployment feedback to reroute subsequent generative trajectories toward more successful behaviors under jointly shifted object and scene distributions.

Table 1: **Adaptation on ManiSkill3.** Online SR@15 and Final SR under object- and scene-level OOD. See § 4.2 for details.

| Agent | SR@15 ↑ | Final SR ↑ |
|---|---|---|
| Base VLA | – | 33.8 |
| RoboMonkey | – | 37.5 |
| AAC | – | 35.7 |
| PDF | 38.6 | 41.8 |
| DSRL | 34.3 | 38.1 |
| **REFLOWTTA** | **49.2**$_{\pm 0.21}$ | **53.6**$_{\pm 0.15}$ |

Table 3: **Adaptation on LIBERO-Plus.** Average SR@20 and Final SR (%) are reported under compound OOD, with LIBERO-Plus perturbations combined with unseen robot embodiments. Further details in § 4.2.

| Agent | SR@20 (%) ↑ | Final SR (%) ↑ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Layout | Camera | R. Init. | Language | Lighting | Background | Noise | Average |
| **Franka Panda** | | | | | | | | | |
| Base VLA | – | 78.4 | 73.6 | 80.8 | 96.2 | 94.1 | 89.0 | 84.5 | 84.6 |
| **UR5e** | | | | | | | | | |
| Base VLA | – | 14.4 | 2.4 | 1.9 | 7.7 | 12.6 | 9.5 | 1.3 | 6.7 |
| RoboMonkey | – | 20.7 | 3.5 | 2.7 | 11.1 | 17.3 | 13.7 | 2.1 | 9.6 |
| AAC | – | 19.3 | 3.7 | 3.1 | 10.3 | 15.8 | 12.8 | 2.6 | 9.2 |
| PDF | 9.7 | 22.9 | 4.5 | 3.8 | 11.9 | 19.1 | 15.4 | 3.2 | 10.9 |
| DSRL | 9.3 | 21.0 | 4.8 | 4.1 | 11.4 | 17.6 | 14.3 | 3.4 | 10.4 |
| **REFLOWTTA** | **14.9**$_{\pm 0.24}$ | **38.6**$_{\pm 0.25}$ | **8.3**$_{\pm 0.31}$ | **6.8**$_{\pm 0.22}$ | **20.5**$_{\pm 0.26}$ | **31.8**$_{\pm 0.18}$ | **27.9**$_{\pm 0.29}$ | **5.1**$_{\pm 0.37}$ | **18.8**$_{\pm 0.16}$ |
| **KUKA IIWA** | | | | | | | | | |
| Base VLA | – | 4.2 | 1.9 | 1.1 | 1.3 | 3.3 | 2.7 | 0.6 | 2.1 |
| RoboMonkey | – | 8.5 | 3.8 | 2.8 | 3.2 | 6.5 | 5.8 | 1.9 | 4.5 |
| AAC | – | 7.6 | 3.5 | 2.9 | 3.0 | 5.9 | 5.2 | 2.1 | 4.2 |
| PDF | 5.1 | 11.1 | 4.8 | 3.7 | 4.1 | 8.2 | 7.2 | 2.7 | 5.8 |
| DSRL | 4.8 | 9.8 | 4.6 | 4.0 | 4.0 | 7.3 | 6.5 | 2.9 | 5.4 |
| **REFLOWTTA** | **9.6**$_{\pm 0.27}$ | **26.6**$_{\pm 0.19}$ | **7.5**$_{\pm 0.28}$ | **6.1**$_{\pm 0.33}$ | **8.4**$_{\pm 0.26}$ | **19.1**$_{\pm 0.22}$ | **17.4**$_{\pm 0.17}$ | **4.9**$_{\pm 0.30}$ | **12.3**$_{\pm 0.15}$ |

Table 4: **Real-world OOD adaptation on PiPER and unseen ARX X5.** SR@15 (%) and Final SR (%) are reported without (top) and with (bottom) human intervention. Further details are provided in § 4.3.

| Agent | PiPER | | ARX X5 | |
|---|---|---|---|---|
| | SR@15 (%) ↑ | Final SR (%) ↑ | SR@15 (%) ↑ | Final SR (%) ↑ |
| *Task Feedback* | | | | |
| Base VLA | – | 32.4 | – | 20.4 |
| RoboMonkey | – | 40.1 | – | 26.6 |
| AAC | – | 34.8 | – | 22.2 |
| PDF | 40.9 | 44.3 | 30.2 | 32.9 |
| DSRL | 35.7 | 38.5 | 25.4 | 28.7 |
| **REFLOWTTA** | **51.6** | **58.6** | **46.1** | **50.7** |
| *Human Intervention* | | | | |
| Base VLA w/ Corrective Data | – | 32.7 | – | 21.8 |
| RICL | – | 48.1 | – | 42.6 |
| **REFLOWTTA** | **57.3** | **62.9** | **53.5** | **58.3** |

**LIBERO: Cross-Embodiment OOD.**For each task,the adaptation budget is 15 episodes, with evaluation on 50 rollouts. As shown in Table 2, replacing the training-time Panda with UR5e or KUKA IIWA reduces the base VLA's Final SR from 96.9% to 20.3% and 9.4%, respectively. REFLOWTTA raises Final SR to **41.0%** and **30.0%**, outperforming PDF (Zang et al., 2026), the strongest baseline, by **14.1%** and **13.6%**. Similar gains are observed in Online SR@15 (**33.9%** *vs* 24.9% and **24.3%** *vs* 15.1%), demonstrating more effective adaptation throughout deployment. These results demonstrate the effectiveness of source-distribution rerouting for adaptation to unseen robot embodiments.

**LIBERO-Plus: Compound OOD.** After 20 deployment episodes per task, Final SR is reported as the variant-weighted average across seven perturbation categories covering 10,030 test variants. Table 3 shows that compound OOD substantially degrades the base VLA, with Final SR falling to 6.7% on UR5e and 2.1% on KUKA IIWA. REFLOWTTA improves Final SR to **18.8%** and **12.3%**, outperforming the strongest baseline, PDF (Zang et al., 2026), by **7.9%** and **6.5%**, respectively. It also achieves higher Online SR@20 (**14.9%** *vs* 9.7% and **9.6%** *vs* 5.1%), indicating more effective adaptation during deployment. These gains support the versatility of source-distribution rerouting in adapting to heterogeneous perturbations combined with embodiment shifts. More results in C.3.

### 4.3 REAL-WORLD EXPERIMENTS

**Setup and Protocol.** The real-world evaluation is conducted on two 6-DoF robotic platforms, AgileX PiPER (Fig. 3) and ARX X5, with the base VLA trained exclusively on PiPER using fixed object–target pairings. During deployment, OOD conditions arise from two sources. **First**, the agent encounters novel object–target combinations, distractors, and unseen objects while remaining on the PiPER embodiment. **Second**, the same frozen VLA is directly transferred to the previously unseen ARX X5, introducing an additional cross-embodiment shift in robot morphology and kine-

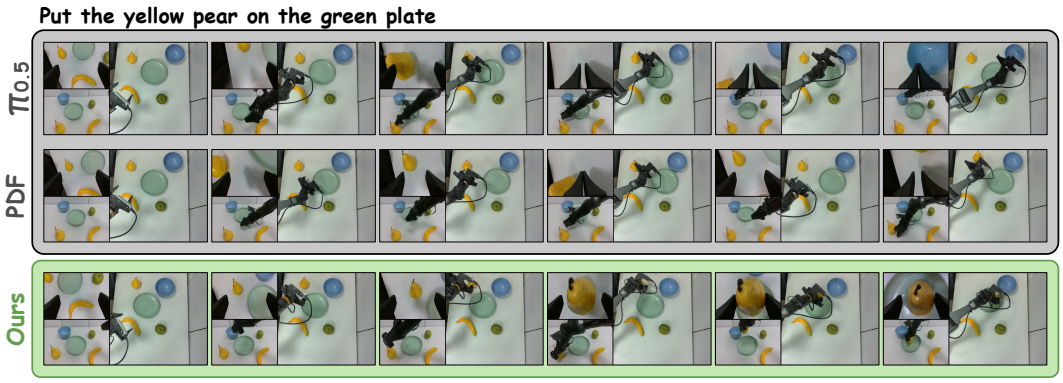



Figure 4: **Qualitative real-world rollouts.** From top to bottom: the base VLA, PDF (Zang et al., 2026), and REFLOWTTA, executing the same manipulation task under the same OOD setting. See § 4.3 for more details.

matics. Evaluation is conducted over 20 real-world tasks with a 15-episode adaptation budget, using 50 rollouts per task after each selected adaptation episode. Further details are in Appendix C.4.

**Quantitative Results.** Table 4 evaluates REFLOWTTA under two settings: adaptation using only environment feedback (§ 3.2), and adaptation with additional human intervention (§ 3.3). For a fair comparison in the latter setting, the same human-intervention data are provided to RICL (Sridhar et al., 2025), which adapts from task demonstrations, and used to fine-tune the base VLA together with its original training data. Under task feedback, REFLOWTTA outperforms all baselines on both embodiments. On PiPER, it exceeds the strongest baseline PDF (Zang et al., 2026) by **10.7%** in SR@15 and **14.3%** in Final SR; on the unseen ARX X5, the margins further increase to **15.9%** and **17.8%**, respectively. These results demonstrate that source-distribution rerouting can effectively exploit deployment feedback across both the original and unseen embodiments. Incorporating human intervention further improves SR@15 and Final SR on PiPER to **57.3%** and **62.9%**, respectively, while achieving a **14.8%** higher Final SR than RICL (Sridhar et al., 2025). By comparison, directly fine-tuning the base VLA with the human-intervention data provides little improvement under the small intervention budget. By mapping these sparse corrections back to the low-dimensional source space, REFLOWTTA can exploit limited human feedback more efficiently than direct VLA fine-tuning or in-context adaptation.

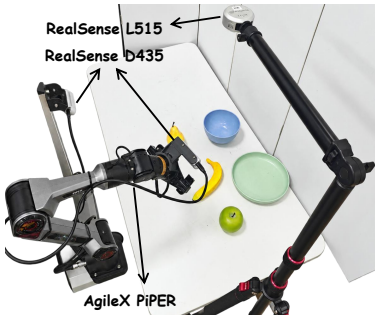



Figure 3: **Real-world setup.**

**Qualitative Visualization.** Fig. 4 shows representative rollouts from the evaluation. The base VLA rigidly continues the subsequent execution after missing the target object, eventually moving toward an incorrect placement region, while PDF (Zang et al., 2026) repeatedly attempts to grasp the object but fails to complete a successful pickup. In contrast, REFLOWTTA reliably grasps the target object and completes the intended placement. These qualitative results further show that adapting the low-dimensional source distribution can reroute the frozen VLA away from repeated failure modes toward task-consistent behavior under deployment. More qualitative results are in Appendix C.4.

### 4.4 ANALYSIS OF TEST-TIME ADAPTATION

**Adaptation Dynamics and Stabilization.** To characterize adaptation dynamics and test whether the learned source distribution remains in a favorable region rather than drifting thereafter, we extend adaptation from 15 to 30 deployment episodes across ten ManiSkill3 tasks from § 4.2 and track the mean SR throughout adaptation. For REFLOWTTA, we additionally track the source update magnitude $\|\boldsymbol{\mu}_\phi^{(e)} - \boldsymbol{\mu}_\phi^{(e-1)}\|_2$, where $\boldsymbol{\mu}_\phi^{(e)}$ denotes the source mean after episode $e$.

As shown in Fig. 5, REFLOWTTA improves SR rapidly within the first 15 deployment episodes and then reaches a highest plateau. By contrast, the other baselines improve more slowly, either leveling off at a lower SR around episode 20 or showing no clear plateau even over the extended 30-episode horizon. Consistent with the higher Online SR@15 in Table 1, this early gain suggests that its low-dimensional source update enables effective adaptation from limited deployment feedback. Beyond episode 15, its mean SR remains stable and its SD narrows. The source-update magnitude rises

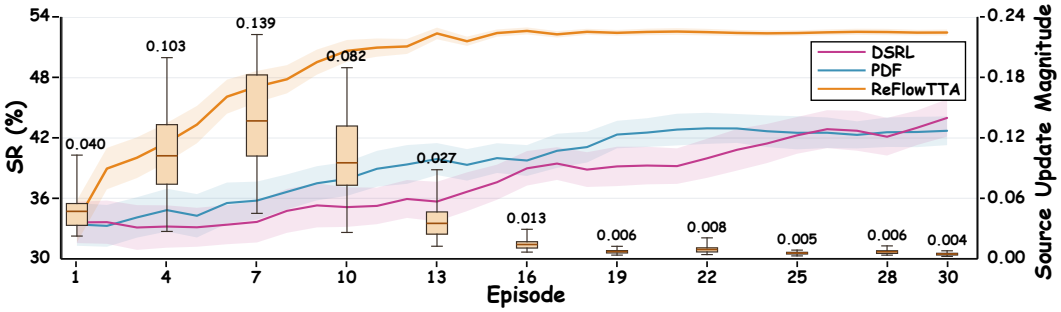



Figure 5: **Test-time adaptation dynamics and stabilization.** Curves show the mean SR across ten ManiSkill3 tasks over 30 deployment episodes, with shading denoting the standard deviation (SD) across tasks. Box plots show the ReFlowTTA source update magnitude, with medians labeled above each box. See § 4.4 for details.

during early improvement, then drops sharply and stays small as SR plateaus, suggesting that the learned source distribution remains in a favorable region without appreciable drift.

**Source-Space Dimensionality.** To examine how the dimension $d$ of the learnable source coordinate $\boldsymbol{m}_t$ affects adaptation with limited deployment feedback, $d$ is varied in the ManiSkill3 OOD setting described in § 4.2, ranging from highly compressed subspaces to the full source space.

As shown in Table 5, overly small source spaces provide insufficient degrees of freedom to represent the source variations needed to effectively reroute the flow trajectory, leaving adaptation largely ineffective. Increasing $d$ initially improves adaptation by expanding this representational capacity, whereas the benefit diminishes as the source space becomes excessively large. Under limited test-time feedback, a higher-dimensional source distribution introduces more degrees of freedom that must be identified and optimized from only a small number of deployment interactions, making reliable adaptation increasingly difficult. These results reveal a dimensionality trade-off between expressive flexibility for rerouting the flow and reliable adaptation from sparse deployment feedback, with $d = 64$ providing the best balance.

Table 5: **Effect of source-space dimensionality.** SR@15 and Final SR across different dimensions $d$.

| $d$ | SR@15 ↑ | Final SR ↑ |
|---|---|---|
| 8 | 33.6 | 33.4 |
| 16 | 35.8 | 36.2 |
| 32 | 46.5 | 48.1 |
| **64** | **49.2** | **53.6** |
| 128 | 47.4 | 49.3 |
| 320 (Full) | 41.6 | 44.8 |

**Directionality of Source-Space Adaptation.** To examine whether the adapted source distribution captures deployment-informed structure beyond the effect of generic source perturbations, the source distribution learned under the ManiSkill3 OOD setting in § 4.2 is modified and re-evaluated under the same protocol. Let $\Delta\boldsymbol{\mu}_\phi^{(0\to15)}$ denote the source-mean displacement over the 15 adaptation episodes, with $r = \|\Delta\boldsymbol{\mu}_\phi^{(0\to15)}\|_2$. Based on this displacement, four variants selectively alter its direction and/or magnitude. The first preserves the learned direction while replacing its magnitude with $\rho \sim \mathrm{Unif}(0, r)$ (**RM**, random magnitude); the second preserves the learned magnitude $r$ but replaces its direction with a random unit vector $\boldsymbol{u} \sim \mathrm{Unif}(\mathbb{S}^{d-1})$ (**RD**, random direction); the third reverses this displacement while keeping its magnitude unchanged (**OD**, opposite direction); and the fourth randomizes both direction and magnitude as $\rho\boldsymbol{u}$ (**RDM**, random direction and magnitude).

Table 6: **Source displacement ablations.** Final SR under different source modifications.

| Variant | Final SR ↑ |
|---|---|
| RM | 41.3 |
| RD | 34.5 |
| OD | 31.9 |
| RDM | 33.2 |
| **Learned** | **53.6** |

In Table 6, preserving the learned direction retains considerable adaptation gain, whereas perturbations that disrupt or reverse this direction perform close to, or below, the base VLA. This indicates that the improvement is not explained by generic source perturbation or displacement magnitude alone, but by deployment-informed directional structure captured in the learned source shift.

## 5 Conclusion

This work proposes ReFlowTTA, a source-space test-time adaptation framework for VLA agents under heterogeneous OOD conditions. By learning a low-dimensional source distribution from deployment feedback, ReFlowTTA reroutes flow-based action generation along alternative trajectories that more reliably realize pretrained capabilities, providing a versatile adaptation interface across diverse deployment shifts. Extensive simulation and real-world evaluations demonstrate consistent gains in online adaptation and final task performance, validating source-space rerouting as an effective and versatile mechanism for test-time adaptation of VLA agents.

### AI USE STATEMENT

Generative AI tools were used for language polishing, grammar checking, and code debugging. They were also used to explore alternative visual representations and layouts for figures in the paper to improve presentation clarity. All AI-assisted content was reviewed by the authors, who take responsibility for the final content of this work.

### REPRODUCIBILITY STATEMENT

The formulation and optimization procedure of REFLOWTTA are described in § 3, with the complete test-time adaptation procedure summarized in pseudocode in Appendix A. Detailed evaluation protocols are provided in Appendix C, with hyperparameter settings in Appendix C.5 and reward definitions in Appendix C.6. The core implementation is available in an anonymous repository.

## REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Conference on Robot Learning*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *CoRL*, 2025a.

Kevin Black, Noah Brown, Danny Driess, et al. $\pi_0$: A vision-language-action flow model for general robot control. In *RSS*, 2025b.

Yuhui Chen, Shuai Tian, Shugao Liu, et al. ConRFT: A reinforced fine-tuning method for VLA models via consistency policy. In *RSS*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, et al. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Haoyuan Deng, Wenkai Guo, Qianzhun Wang, Zhenyu Wu, and Ziwei Wang. SafeBimanual: Diffusion-based trajectory optimization for safe bimanual manipulation. In *CoRL*, 2025.

Sombit Dey, Jan-Nico Zaech, Nikolay Nikolov, Luc Van Gool, and Danda Pani Paudel. ReVLA: Reverting visual domain limitation of robotic foundation models. In *ICRA*, 2025.

Zibin Dong, Yicheng Liu, Yinchuan Li, et al. Conditioning matters: Training diffusion policies is faster than you think. In *NeurIPS*, 2025.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. In *NeurIPS*, 2024.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *CVPR*, 2026.

Dibya Ghosh, Homer Walke, Karl Pertsch, et al. Octo: An open-source generalist robot policy. In *RSS*, 2024.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, et al. Test time adaptation via conjugate pseudo-labels. In *NeurIPS*, 2022.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *ICLR*, 2019.

Huanlei Guo, Hongxin Wei, and Bingyi Jing. Toward early quality assessment of text-to-image diffusion models. In *CVPR*, 2026.

Xiefan Guo, Jinlin Liu, Miaomiao Cui, et al. Initno: Boosting text-to-image diffusion models via initial noise optimization. In *CVPR*, 2024.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *ICLR*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. In *RSS*, 2024.

Jiaheng Hu, Rose Hendrix, Ali Farhadi, et al. FLaRe: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *ICRA*, 2025a.

Jinwu Hu, Zitian Zhang, Guohao Chen, et al. Test-time learning for large language models. In *ICML*, 2025b.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *CoRL*, 2024.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *RSS*, 2021.

Jacky Kwok, Christopher Agia, Rohan Sinha, et al. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *CoRL*, 2025.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *RSS*, 2026a.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *CVPR*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, et al. Flow matching for generative modeling. In *ICLR*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Changyu Liu, Yiyang Liu, Taowen Wang, et al. On-the-fly vla adaptation via test-time reinforcement learning. In *ACL*, 2026.

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. In *NeurIPS*, 2025a.

Songming Liu, Lingxuan Wu, Bangguo Li, et al. RDT-1B: A diffusion foundation model for bimanual manipulation. In *ICLR*, 2025b.

Nanye Ma, Shangyuan Tong, Haolin Jia, et al. Scaling inference time compute for diffusion models. In *CVPR*, 2025.

Boming Miao, Chunxiao Li, Xiaoxiao Wang, et al. Noise diffusion for enhancing semantic faithfulness in text-to-image synthesis. In *CVPR*, 2025.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, 2022.

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. In *ICRA*, 2024.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, et al. Diffusion policy policy optimization. In *ICLR*, 2025.

Dvir Samuel, Rami Ben-Ari, Simon Raviv, Nir Darshan, and Gal Chechik. Generating images of rare concepts using pre-trained diffusion models. In *AAAI*, 2024.

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

Haoming Song, Delin Qu, Yuanqi Yao, et al. FM-Steer: Enhance generalist policies with value-guided cascaded denoising. In *CVPR*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, et al. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021b.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In *CoRL*, 2025.

Alane Suhr and Yoav Artzi. Continual learning for instruction following from realtime feedback. In *NeurIPS*, 2023.

Yu Sun, Xiaolong Wang, Zhuang Liu, et al. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Stone Tao, Fanbo Xiang, Arth Shukla, et al. Demonstrating gpu parallelized robot simulation and rendering for generalizable embodied ai with maniskill3. In *RSS*, 2025.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, et al. Steering your diffusion policy with latent space reinforcement learning. In *CoRL*, 2025.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, et al. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic generalist policy distillation via reinforcement learning. In *RSS*, 2025.

Takuma Yoneda, Ge Yang, Matthew R. Walter, and Bradly C. Stadie. Invariance through latent alignment. In *RSS*, 2022.

Linjing You, Jiabao Lu, and Xiayuan Huang. Test-time correlation alignment. In *ICML*, 2025.

Chao Yu, Yuanqing Wang, Zhen Guo, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *OSDI*, 2026.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2017.

Xiu Yuan, Tongzhou Mu, Stone Tao, et al. Policy decorator: Model-agnostic online refinement for large policy model. In *ICLR*, 2025.

Zehua Zang, Xi Wang, Fuchun Sun, Xiao Xu, Lixiang Liu, Jiahuan Zhou, and Jiangmeng Li. Test-time perturbation tuning with delayed feedback for vision-language-action models. In *CVPR*, 2026.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *NeurIPS*, 2022.

Yue Zhang, Rui Wang, Jiehong Lin, Zhongrui Wang, and Xiaojuan Qi. Retrieval-VLA: Training-Free In-Context Adaptation for Vision-Language-Action Models. In *CVPR*, 2026.

Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, et al. Golden noise for diffusion models: A learning framework. In *ICCV*, 2025.

Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. In *CoRL*, 2023.

# APPENDIX CONTENTS


**A Pseudo Code** **14**

**B Diagnostic Analyses of OOD Failures** **15**

B.1 Behavioral Structure of Failed OOD Rollouts . . . . . 15

B.2 Task Performance across Denoising Trajectories . . . . . 15

**C Implementation Details** **16**

C.1 ManiSkill3 Evaluation Details. . . . . 16

C.2 LIBERO Evaluation Details . . . . . 17

C.3 LIBERO-Plus Evaluation Details . . . . . 17

C.4 Real-World Evaluation Details . . . . . 18

C.5 Hyperparameter Settings and Sensitivity Analysis . . . . . 20

C.6 Reward Function . . . . . 21

C.7 Baseline Implementation Details . . . . . 21

C.8 Codebase . . . . . 22

**D Failure Cases and Scope of Adaptation** **22**

**E Supplementary Videos** **22**

**F Limitation and Future Work** **23**

# A PSEUDO CODE

Algorithm 1 summarizes the overall deployment-time adaptation procedure of REFLOWTTA. The pretrained VLA $\pi_\theta$ and the adaptation basis $\boldsymbol{B}$ remain frozen throughout deployment, while only the source-distribution mean $\boldsymbol{\mu}_\phi$ is optimized. Human corrections, when available, are mapped back to the same source space and incorporated into the adaptation objective.

**Algorithm 1** Pseudo-code of REFLOWTTA in a PyTorch-like style.

```
# pi_theta: frozen VLA; B: fixed orthonormal basis
# mu_phi: learnable source mean; I_d: identity covariance
# D_u: rollout buffer; H_u: recovered human source coordinates
# K_roll: autonomous plus correction chunks per update
# T_e: executed actions per chunk
# JOINT_LOG_PROB sums log-probabilities over source dimensions

def BUILD_SOURCE(eps_t, m_t, B):
    #========== Source construction (Eq. 3) ==========#
    e_t = VEC(eps_t)
    return VEC_INV((I - B @ B.T) @ e_t + B @ m_t)

def UPDATE_SOURCE(D_u, H_u, mu_phi):
    q_phi = GAUSSIAN(mu_phi, I_d)
    #========== Chunk-level PPO (Eqs. 6, 7) ==========#
    L_src = 0
    if SIZE(D_u) > 0:
        for m_t, logp_old, r_t in D_u:
            rho_t = EXP(JOINT_LOG_PROB(q_phi, m_t) - logp_old)
            rho_clip = CLIP(rho_t, 1-eps_clip, 1+eps_clip)
            L_src -= MIN(rho_t * r_t, rho_clip * r_t)
        L_src = L_src / SIZE(D_u)

    #========== Human source loss (Eq. 10) ==========#
    L_H = 0
    if SIZE(H_u) > 0:
        for m_H in H_u:
            L_H -= JOINT_LOG_PROB(q_phi, m_H)
        L_H = L_H / SIZE(H_u)
    loss = L_src + lambda_H * L_H  # Eq. 11
    return OPTIMIZER_STEP(mu_phi, loss)  # update mu_phi only

def REFLOWTTA(pi_theta, B):
    mu_phi = ZEROS(d)  # initialize once for each OOD condition
    # Retain mu_phi across episodes under the same condition.
    while DEPLOYMENT_CONTINUES():
        q_old = GAUSSIAN(FROZEN_COPY(mu_phi), I_d)
        D_u, H_u = [], []

        #========== Collect K_roll feedback-bearing chunks ==========#
        while SIZE(D_u) + SIZE(H_u) < K_roll:
            c_t = OBSERVE_CONDITION()
            if CORRECTION_AVAILABLE(c_t):
                A_H_env = GET_CORRECTION(c_t)  # full [T, 7] chunk
                A_H = ENCODE_ACTION(A_H_env)   # normalized [T, D]
                eta_H = INVERSE_FLOW(A_H, c_t)  # Eq. 8
                H_u.append(B.T @ VEC(eta_H))   # Eq. 9
                EXECUTE_PREFIX(A_H_env, T_e)
            else:
                m_t = SAMPLE(q_old)
                eps_t = SAMPLE_STANDARD_NORMAL(T, D)
                eta_t = BUILD_SOURCE(eps_t, m_t, B)
                A_t = pi_theta(c_t, eta_t)
                EXECUTE_PREFIX(A_t, T_e)
                r_t = CHUNK_REWARD()
                logp_old = JOINT_LOG_PROB(q_old, m_t)
                D_u.append((m_t, logp_old, r_t))  # Eq. 4

        #========== Update source distribution only ==========#
        mu_phi = UPDATE_SOURCE(D_u, H_u, mu_phi)
```

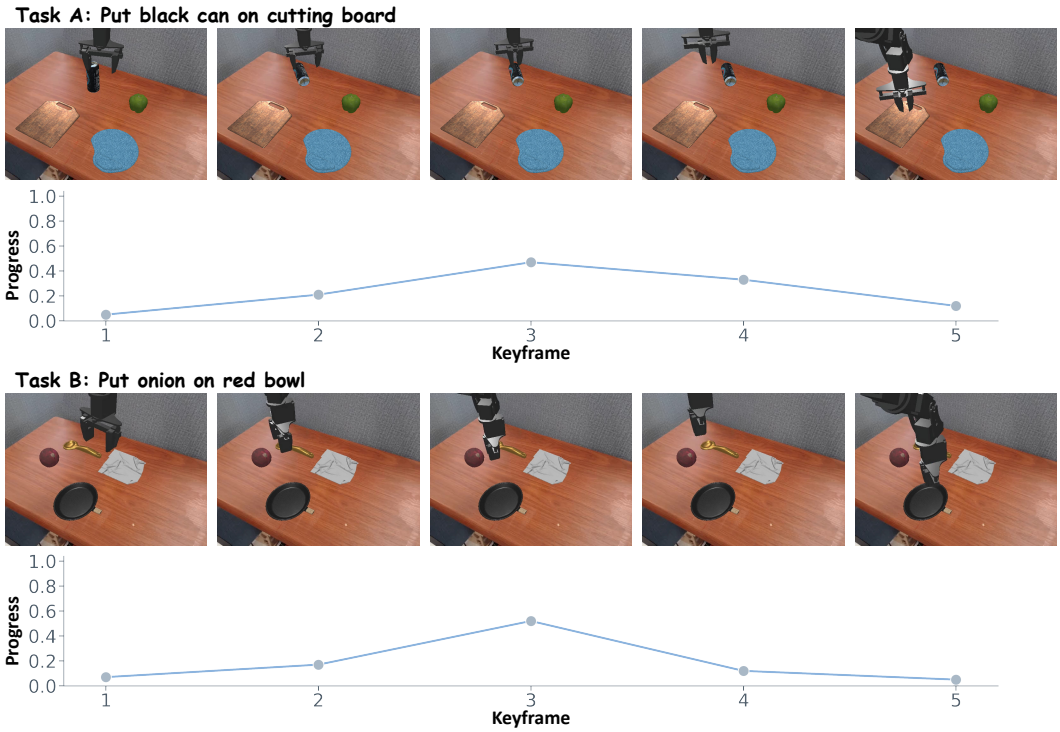



Figure 6: **Qualitative examples of OOD rollouts. Task A (top):** The pretrained agent fails to adapt its grasp to the novel black can. After toppling it, the agent does not regrasp but continues toward the target. **Task B (bottom):** Under the novel scene and object configuration, the agent produces a spatially misaligned grasp yet continues to execute the nominal motion trajectory.

# B Diagnostic Analyses of OOD Failures

To investigate how VLA agents fail under OOD deployment, we conduct two complementary diagnostic experiments that examine, respectively, the behavioral structure of failed rollouts and how task performance varies across denoising trajectories. Experiments are conducted under OOD conditions involving both scene configurations and objects that are absent from the training distribution.

## B.1 Behavioral Structure of Failed OOD Rollouts

Representative OOD rollouts are visualized alongside dense task-progress curves to jointly capture the agent's action sequence and its progress toward task completion. As shown in Fig. 6, the agent in Task A reaches and attempts to manipulate the correct object, but fails to grasp the novel black can properly and continues toward the target after toppling it. A similar pattern appears in Task B, where the gripper closes at an offset position while the subsequent motion continues to follow the intended grasp-and-transfer sequence. These broadly task-relevant behaviors also attain nontrivial intermediate progress before ultimately failing, as reflected by the corresponding progress curves. Taken together, the preserved manipulation targets, task-consistent action sequences, and measurable task progress suggest that OOD failure does not necessarily reflect a complete collapse of task-relevant behavior, but rather an unreliable realization of the capabilities retained by the pretrained agent.

## B.2 Task Performance across Denoising Trajectories

The observation in § B.1 motivates examining whether the capabilities retained by the pretrained agent can be realized through alternative denoising trajectories. Accordingly, the agent is kept frozen while Gaussian perturbations are injected early in the denoising process, inducing alternative denoising trajectories for action generation and probing its reachable behavior space.

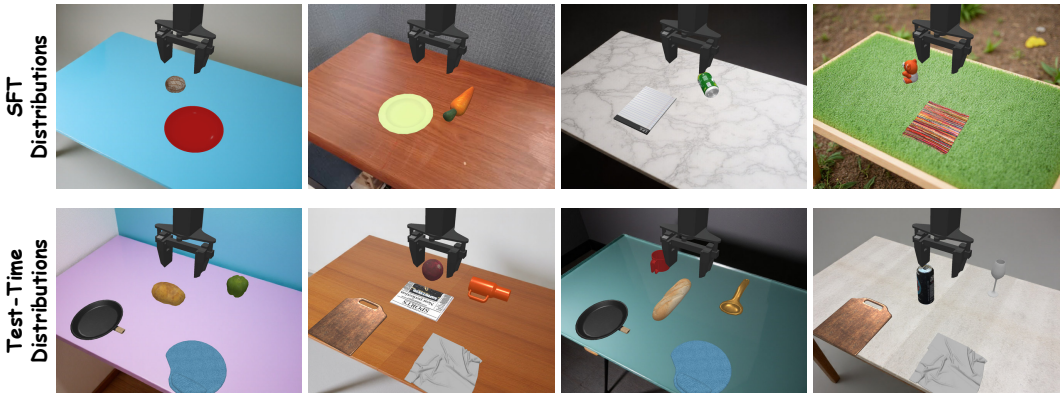



Figure 7: Representative ManiSkill3 scenes from the SFT distribution (top) and the test-time OOD distribution (bottom). Unlike the SFT scenes, test-time scenes jointly introduce held-out object categories and table appearances, finer-grained object poses, and multiple distractor objects.

Let $\boldsymbol{x}_K$ denote the initial source action trajectory, and let $\mathcal{D}_\theta$ denote one denoising update of the frozen agent. The default denoising process follows

$$\boldsymbol{x}_{k-1} = \mathcal{D}_\theta(\boldsymbol{x}_k, k), \qquad k = K, \dots, 1. \tag{12}$$

Alternative denoising trajectories are generated by first perturbing the source state,

$$\tilde{\boldsymbol{x}}_K = \boldsymbol{x}_K + \tau \boldsymbol{\epsilon}_K, \qquad \boldsymbol{\epsilon}_K \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{13}$$

and injecting Gaussian perturbations during the first $M$ denoising steps,

$$\tilde{\boldsymbol{x}}_{k-1} = \mathcal{D}_\theta(\tilde{\boldsymbol{x}}_k, k) + \tau \boldsymbol{\epsilon}_{k-1}, \qquad k = K, \dots, K - M + 1, \tag{14}$$

where

$$\boldsymbol{\epsilon}_{k-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{15}$$

The remaining denoising steps proceed without additional perturbations:

$$\tilde{\boldsymbol{x}}_{k-1} = \mathcal{D}_\theta(\tilde{\boldsymbol{x}}_k, k), \qquad k = K - M, \dots, 1. \tag{16}$$

Here, $\tau$ controls the perturbation magnitude, with $\tau = 0$ recovering the default denoising trajectory and larger values inducing greater deviations from the original path.

Table 7 reports the success rates on Tasks A and B across different perturbation scales, with $K = 4$ denoising steps and perturbations introduced during the first $M = 2$ steps. At $\tau = 0$, all rollouts fail on both tasks, whereas successful rollouts begin to emerge as the perturbation magnitude increases. Although the absolute success rates remain modest, their emergence without any parameter updates suggests that the corresponding successful behaviors are already embedded in the pretrained agent's action manifold. The default denoising flow, however, has limited access to these behaviors under OOD deployment, making the observed failures a matter of behavioral accessibility rather than missing capability.

Table 7: **Task performance under varying perturbation scales.** Success rates (SR), reported as fractions over 50 rollouts per setting, on Tasks A and B under different Gaussian perturbation scales $\tau$.

| Task | Source-noise temperature $\tau$ | | | |
|---|---|---|---|---|
| | **0.0** | **0.3** | **0.6** | **0.9** |
| **Task A** | 0.00 | 0.02 | 0.08 | 0.04 |
| **Task B** | 0.00 | 0.06 | 0.06 | 0.04 |

# C Implementation Details

## C.1 ManiSkill3 Evaluation Details

Relative to SFT, the test-time distribution jointly varies object categories, table appearances, object positions and orientations, and scene composition. We construct 50 OOD pick-and-place tasks under these distribution shifts. Table 8 details these distribution differences, while Fig. 7 shows representative SFT and OOD scenes.

Table 8: **SFT and test-time distributions on ManiSkill3.** All test-time variations are jointly instantiated within each OOD scene rather than evaluated independently.

| Dimension | SFT Distribution | Test-Time OOD Distribution |
|---|---|---|
| **Source objects** | 16 categories: carrot, kitchen shovel, bread, plastic bottle, 7up can, zuchinni, ketchup bottle, watering can, pipe, toy bear, fast food cup, plant, banana, hamburger, golf ball, BBQ sauce | 9 held-out categories: travel cup, pepper, nonstop can, potato, baguette, champagne glass, kitchen spoon, onion, cup |
| **Target objects** | 12 categories: plate, carpet, sheet metal, drawing tablet, tomato slice, pizza, flat bowl, gramophone disk, chess board, manhole cover, envelope, notepad | 5 held-out categories: mouse pad, frying pan, cutting board, newspaper, cloth |
| **Table appearance** | 16 appearances: brown wooden table, blue plastic table, black TV cabinet, light-gray picnic table, light wooden desk, wooden table, yellow desk, white glossy table, gray table, white table, white marble table, pink table, dark-gray table, grass-texture table, dark-brown wooden table, multicolor desk | 5 held-out appearances: glass table, green table, pink desk, stone table, dining table |
| **Object position** | $x \in [-0.235, -0.085]$ m, $y \in [-0.075, 0.075]$ m; $6 \times 6$ grid | $x \in [-0.265, -0.055]$ m, $y \in [-0.105, 0.105]$ m; $8 \times 8$ grid |
| **Object orientation** | Source yaw: $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$; target yaw fixed at $0^\circ$ | Source and target yaw: $\{0^\circ, 15^\circ, \ldots, 345^\circ\}$ |
| **Scene composition** | $1 \times 1$ source–target | $2 \times 2$, $2 \times 3$, or $3 \times 2$ source–target |

Table 9: **Robot configurations for cross-embodiment evaluation.** Franka Panda is used for SFT, while UR5e and KUKA IIWA are introduced during test-time adaptation.

| Robot | DoF | Gripper | Stage |
|---|---|---|---|
| **Franka Panda** | 7 | Panda Gripper | SFT |
| **UR5e** | 6 | Robotiq 2F85 | Test-time |
| **KUKA IIWA** | 7 | Robotiq 2F140 | Test-time |

### C.2 LIBERO Evaluation Details

As summarized in Table 9, the evaluation replaces the Franka Panda used during SFT with a UR5e or KUKA IIWA at test time. These robots differ in arm DoFs and gripper configurations, introducing changes in robot morphology, visual appearance, workspace, and interaction dynamics.

A key challenge in adapting a frozen generative VLA across embodiments is that its action representation and output dimensionality are fixed after pretraining, whereas different robots may expose heterogeneous joint-space action interfaces. To bridge this mismatch without modifying the pretrained policy, the cross-embodiment setting adopts a common Cartesian end-effector action representation:

$$\boldsymbol{a}_{t,j}^{\text{exec}} = [\Delta x, \Delta y, \Delta z, \Delta r_x, \Delta r_y, \Delta r_z, g] \in \mathbb{R}^7, \tag{17}$$

where the first three dimensions specify end-effector translation, the next three specify axis-angle rotation, and $g \in \{0, 1\}$ denotes the gripper open/close command, thereby decoupling the fixed VLA action interface from embodiment-specific joint control while preserving the original action-chunk structure. Here, $\boldsymbol{a}_{t,j}^{\text{exec}}$ consists of the first seven components of the model-predicted action $\boldsymbol{a}_{t,j} \in \mathbb{R}^D$ at decision step $t$ and chunk index $j$.

The proprioceptive input is similarly represented in a common task space:

$$\boldsymbol{q}_t = [\boldsymbol{p}_{\text{ee}}^{(3)}, \boldsymbol{r}_{\text{ee}}^{(3)}, g_{\text{state}}] \in \mathbb{R}^7, \tag{18}$$

where $\boldsymbol{p}_{\text{ee}}$ and $\boldsymbol{r}_{\text{ee}}$ denote the end-effector position and axis-angle orientation, respectively, and $g_{\text{state}} \in \{0, 1\}$ indicates whether the gripper is open or closed.

Across embodiments, the task definitions and success criteria remain unchanged, with success determined by the original LIBERO BDDL predicates.

### C.3 LIBERO-Plus Evaluation Details

LIBERO-Plus follows the cross-embodiment protocol described in Appendix C.2, with the same robot configurations, Cartesian action space, and task-space proprioceptive representation.

For the robot-initial-state perturbation, LIBERO-Plus directly perturbs the initial joint configuration. To accommodate different embodiments, the induced end-effector displacement is obtained via forward kinematics on Panda, transferred to the target embodiment, and mapped back to joint space through inverse kinematics.

Table 10: **Detailed SR@20 on LIBERO-Plus.** SR@20 (%) across the seven perturbation dimensions under the two unseen robot embodiments.

| Robot | Agent | Layout | Camera | Robot Init. | Language | Lighting | Background | Noise | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| UR5e | Base VLA | – | – | – | – | – | – | – | – |
| | RoboMonkey | – | – | – | – | – | – | – | – |
| | AAC | – | – | – | – | – | – | – | – |
| | PDF | 20.3 | 3.9 | 3.3 | 10.7 | 17.1 | 13.8 | 2.7 | 9.7 |
| | DSRL | 18.8 | 4.2 | 3.6 | 10.1 | 15.7 | 12.6 | 3.0 | 9.3 |
| | **REFLOWTTA** | **30.8**±0.25 | **6.5**±0.33 | **5.3**±0.29 | **16.4**±0.26 | **25.1**±0.22 | **21.8**±0.28 | **4.0**±0.35 | **14.9**±0.24 |
| KUKA IIWA | Base VLA | – | – | – | – | – | – | – | – |
| | RoboMonkey | – | – | – | – | – | – | – | – |
| | AAC | – | – | – | – | – | – | – | – |
| | PDF | 9.6 | 4.3 | 3.3 | 3.7 | 7.2 | 6.3 | 2.4 | 5.1 |
| | DSRL | 8.5 | 4.1 | 3.6 | 3.6 | 6.5 | 5.8 | 2.6 | 4.8 |
| | **REFLOWTTA** | **20.9**±0.28 | **5.9**±0.34 | **4.8**±0.31 | **6.5**±0.29 | **14.7**±0.25 | **13.4**±0.30 | **3.8**±0.36 | **9.6**±0.27 |

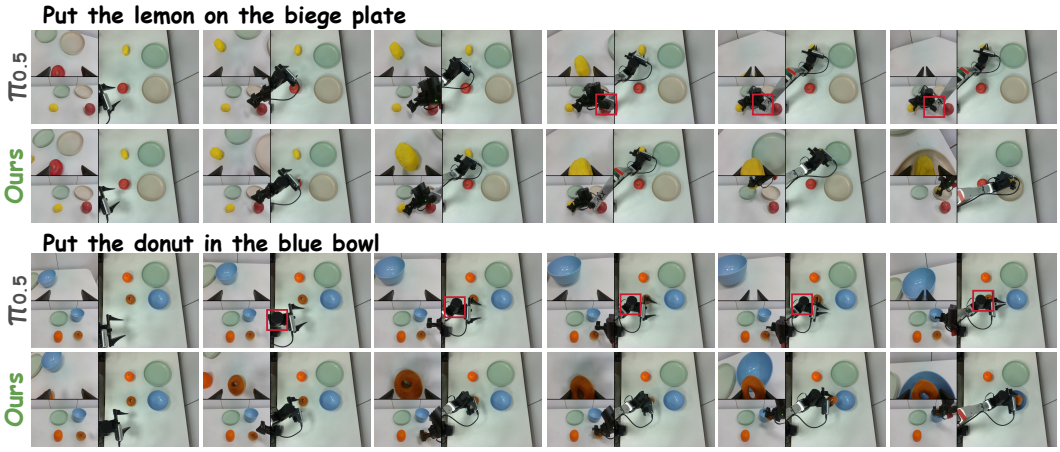



Figure 8: **Qualitative cross-embodiment results.** REFLOWTTA recovers successful behaviors from Base VLA failures caused by embodiment mismatch. Relevant joints are highlighted in red. Details in Appendix C.4.

The 10,030 LIBERO-Plus variants are distributed across Layout, Camera, Robot Initial State, Language, Lighting, Background, and Noise as $(1525, 1599, 1550, 1537, 1142, 1076, 1601)$ (Fei et al., 2026). For both Final SR and SR@20, we report the variant-weighted average $\sum_c n_c \, \mathrm{SR}_c / 10{,}030$.

Table 10 reports SR@20 across the seven perturbation dimensions for the two unseen embodiments. REFLOWTTA achieves an average SR@20 of **14.9%** on UR5e and **9.6%** on KUKA IIWA, compared with 9.7% and 5.1% for PDF, respectively. The higher SR@20 further indicates more effective adaptation throughout deployment under limited interaction feedback.

### C.4 REAL-WORLD EVALUATION DETAILS

Because physical rollouts are time-consuming, we compute real-world Online SR@15 for each agent by averaging its success rates after adaptation episodes 2, 4, 6, 8, 10, 12, 14, and 15.

**Cross-Embodiment Joint-Space Evaluation.** Both the AgileX PiPER and ARX X5 feature six actuated arm joints, allowing the real-world evaluation to use a six-dimensional joint action space as a complement to the end-effector-space control used in simulation (Appendices C.2 and C.3). Operating directly in joint space preserves embodiment-specific mappings from joint actions to physical motion, such that differences in joint-axis definitions and kinematic geometry are directly reflected during cross-embodiment transfer.

Table 11 summarizes the joint limits and rotation axes of the two manipulators, where axis directions are expressed in the robot base frame at the nominal configuration. PiPER and ARX X5 differ in both their feasible joint ranges and joint-axis arrangements, with particularly pronounced discrepancies in the distal wrist joints, where J4 and J5 rotate about different axes across the two embodiments. Beyond these joint-level differences, the two manipulators also differ in their kinematic geometries, including link lengths, joint placements, and fixed inter-joint offsets. Together, these discrepancies

Table 11: **Joint-level differences between AgileX PiPER and ARX X5.** Axis directions are expressed in the robot base frame at the nominal configuration.

| Joint | PiPER | | ARX X5 | |
|---|---|---|---|---|
| | Range | Axis | Range | Axis |
| J1 | $[-150°, 150°]$ | $+Z$ | $[-150°, 180°]$ | $+Z$ |
| J2 | $[0°, 180°]$ | $+Y$ | $[0°, 210°]$ | $+Y$ |
| J3 | $[-170°, 0°]$ | $+Y$ | $[0°, 180°]$ | $-Y$ |
| J4 | $[-100°, 100°]$ | $+X$ | $[-90°, 90°]$ | $-Y$ |
| J5 | $[-70°, 70°]$ | $+Y$ | $[-90°, 90°]$ | $-Z$ |
| J6 | $[-120°, 120°]$ | $+X$ | $[-120°, 120°]$ | $+X$ |

yield embodiment-specific forward kinematics and joint-to-motion mappings, such that the same joint-space action can produce substantially different physical motions across the two robots.

To enable the agent to explore from the action-generation regime already learned on PiPER, trivial coordinate-level discrepancies arising from different joint conventions are mitigated before deploying the policy on ARX X5. Specifically, fixed joint-wise offsets are applied according to the difference between approximately corresponding initial configurations, such that the joint-state conditioning during denoising remains close to the coordinate region covered during training on PiPER. Let $\boldsymbol{q}^A$ denote the native ARX joint coordinates and $\tilde{\boldsymbol{q}}^A$ the aligned coordinates used for policy conditioning. Their relationship is defined as

$$\tilde{\boldsymbol{q}}^A = \boldsymbol{S}\boldsymbol{q}^A + \boldsymbol{b}, \tag{19}$$

where

$$\boldsymbol{S} = \mathrm{diag}(1, 1, -1, 1, 1, 1), \qquad \boldsymbol{b} = [0°,\ 0°,\ 0°,\ 50°,\ 40°,\ 0°]. \tag{20}$$

For J3, the native joint ranges of PiPER and ARX X5 have no non-zero overlap and follow opposite coordinate directions; its sign is therefore reversed through $\boldsymbol{S}$ to establish a source-compatible joint convention. Since the policy predicts joint-angle displacements, the corresponding ARX action is mapped back through the same sign transformation,

$$\Delta\boldsymbol{q}^A = \boldsymbol{S}\Delta\tilde{\boldsymbol{q}}^A. \tag{21}$$

Although this alignment reduces part of the cross-embodiment OOD, it only mitigates trivial coordinate mismatches and does not alter the physical embodiment. Substantial differences in joint-axis orientations, link geometry, joint limits, and kinematics therefore remain for the agent to accommodate during deployment.

**Qualitative Cross-Embodiment Analysis.** Fig. 8 further visualizes how the embodiment gap described above alters the physical effect of joint-space actions, and how REFLOWTTA adapts the generated motion to recover successful behavior on the target robot. In the first case, the agent trained on PiPER uses J4 to adjust the gripper orientation in roll during grasping, following the coordinate convention in Tab. 11. On ARX X5, however, J4 rotates along the pitch direction, causing the same motion tendency to lift the end effector upward rather than primarily adjust the grasp orientation. As a result, the Base VLA approaches above the target object and misses the grasp. After adaptation, REFLOWTTA reduces the J4 motion while making greater use of the J2 and J3 joints closer to the base to lower and extend the end-effector trajectory, recovering a successful grasp.

In the second case, the agent uses J5 to rotate the gripper in pitch, tilting it downward toward the target during the grasp approach, whereas J5 on ARX X5 rotates along the yaw direction. Combined with differences in link geometry, this changes the end-effector motion into a pronounced lateral deviation from the intended grasp pose, causing the Base VLA to miss the object. REFLOWTTA reduces the excessive J5 motion, limiting the lateral deviation and restoring the gripper–object alignment required for successful grasping. Together, these cases show that REFLOWTTA can compensate for embodiment-dependent changes in how joint-space actions translate into physical motion, rerouting action generation toward successful trajectories compatible with the target robot.

**Real-World Adaptation Process.** Fig. 9 illustrates the behavioral progression of REFLOWTTA during real-world deployment. The frozen $\pi_{0.5}$ retains the coarse task structure but fails to grasp the banana, subsequently continuing its motion toward the target bowl despite the failed grasp. As test-time adaptation proceeds, the agent gradually exhibits more effective grasp behavior. During the seventh deployment episode of the 15-episode adaptation process, the agent repeatedly re-approaches

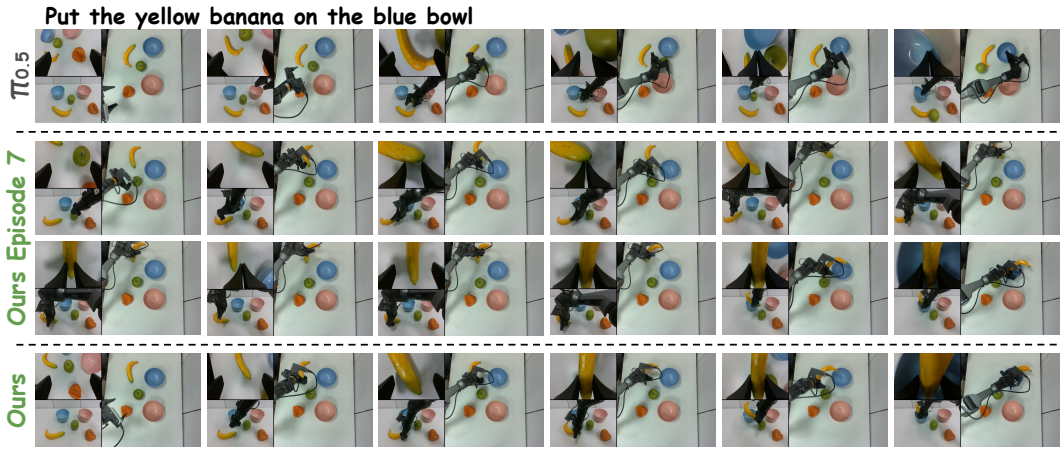



Figure 9: **Qualitative illustration of the adaptation process. Top:** The frozen $\pi_{0.5}$ fails to grasp the banana. **Middle:** At Episode 7, the agent tries multiple grasp angles before eventually succeeding. **Bottom:** After adaptation, the agent follows a previously successful grasp pattern and succeeds on its first attempt. More details are provided in Appendix C.4.

the banana and adjusts its grasp direction through feedback-driven updates, eventually establishing a successful grasp. At this intermediate stage, a successful action-generation trajectory has become accessible, although successful execution still requires multiple attempts. After completing all 15 deployment episodes, the final agent successfully grasps the banana on its first attempt. Notably, the grasp region selected by the final agent is similar to the region where a successful grasp first emerged after repeated adjustments in Episode 7. This behavioral progression shows how deployment feedback can first expose effective action-generation trajectories and subsequently reinforce them through continued adaptation, transforming a behavior that initially requires repeated adjustments into a more direct and reliable execution.

**Reverse Flow for Human Corrections.** Each corrective action chunk is paired with its conditioning $\boldsymbol{c}_t$. The model's action input transform normalizes and pads the seven-dimensional robot actions to $D = 32$, preserving the action horizon $T$. We integrate the pretrained, frozen velocity field in Eq. 8 with ten explicit Euler steps ($\Delta s = 0.1$), compared with four steps for forward action generation:

$$\boldsymbol{X}^{\mathrm{H}}_{t,j+1} = \boldsymbol{X}^{\mathrm{H}}_{t,j} + \Delta s\, \boldsymbol{v}_\theta\big(\boldsymbol{X}^{\mathrm{H}}_{t,j}, j\Delta s \mid \boldsymbol{c}_t\big), \qquad \boldsymbol{X}^{\mathrm{H}}_{t,0} = \boldsymbol{A}^{\mathrm{H}}_t, \quad j = 0, \ldots, 9. \tag{22}$$

The endpoint $\boldsymbol{X}^{\mathrm{H}}_{t,10}$ estimates $\boldsymbol{\eta}^{\mathrm{H}}_t$ and is projected onto the fixed source basis by Eq. 9. At each decision step, an autonomous transition enters $\mathcal{D}_u$ or a correction enters $\mathcal{H}_u$; each counts as one chunk. Once their combined count reaches $K_{\mathrm{roll}}$, we update $\boldsymbol{\mu}_\phi$ using the available terms of Eq. 11. Autonomous transitions supply the PPO term, while corrections supply the intervention term without entering the PPO probability ratio. If either buffer is empty, its corresponding loss term is omitted.

### C.5 Hyperparameter Settings and Sensitivity Analysis

**Default Hyperparameter Settings.** ReFlowTTA uses a source-space dimension of $d = 64$, which provides the best balance between adaptation flexibility and optimization reliability (§ 4.4). For each task and independent adaptation run, a new orthonormal basis $\boldsymbol{B}$ is randomly sampled and kept fixed throughout the corresponding $K$-episode adaptation. The base VLA used in all experiments is $\pi_{0.5}$, with an action horizon of $T = 10$ and an execution horizon of $T_e = 5$. Its model-side action dimension is $D = 32$, yielding a full source dimension of $TD = 320$. For the 7-dimensional robot control interface used in the experiments, the first seven dimensions of each predicted action are used as the executable command, while the remaining dimensions correspond to the padded action representation of $\pi_{0.5}$. At each decision step, only the first five actions of the predicted chunk are executed before obtaining a new observation and replanning.

A source-mean update is performed after collecting $K_{\mathrm{roll}} = 4$ chunks, counting both autonomous transitions and human corrections when available. The learning rate for optimizing the source-

distribution mean $\boldsymbol{\mu}_\phi$ is set to $1\times10^{-3}$, and the PPO clipping parameter is $\varepsilon_{\text{clip}} = 0.2$, corresponding to a probability-ratio clipping interval of $[0.8, 1.2]$.

**Sensitivity Analysis.** The experiments are conducted on ManiSkill3 OOD using the same adaptation and evaluation protocol as in § 4.2. It evaluates the sensitivity of REFLOWTTA to the number of chunk-level transitions per update $K_{\text{roll}}$, the learning rate for $\boldsymbol{\mu}_\phi$, the PPO clipping parameter $\varepsilon_{\text{clip}}$, and the number of actions executed from each predicted chunk. Each hyperparameter is varied individually while all other settings are fixed to their default values. Specifically, $K_{\text{roll}}$ is varied over $\{3, 4, 5, 6\}$, the learning rate over $\{3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}\}$, and $\varepsilon_{\text{clip}}$ over $\{0.1, 0.2, 0.3\}$. For chunk execution, the number of executed actions is varied over $\{3, 4, 5, 6, 7\}$ while keeping the predicted action horizon fixed at $T = 10$. As shown in Table 12, the adaptation performance varies only modestly over the tested ranges of all four hyperparameters. This indicates that the adaptation gains do not rely on narrowly tuned hyperparameter choices.

Table 12: **Hyperparameter Sensitivity Analysis.** SR@15 and Final SR (%) under one-at-a-time hyperparameter variations. See § C.5 for details.

| Hyperparameter | Value | SR@15 (%)↑ | Final SR (%)↑ |
|---|---|---|---|
| $K_{\text{roll}}$ | 3 | 48.7 | 52.9 |
| | 4 | 49.2 | 53.6 |
| | 5 | **49.6** | **53.9** |
| | 6 | 48.8 | 52.7 |
| Learning rate | $3 \times 10^{-4}$ | 48.5 | 52.6 |
| | $1 \times 10^{-3}$ | 49.2 | **53.6** |
| | $3 \times 10^{-3}$ | **49.7** | 53.1 |
| $\varepsilon_{\text{clip}}$ | 0.1 | 48.6 | 52.5 |
| | 0.2 | 49.2 | 53.6 |
| | 0.3 | **49.4** | **53.8** |
| Executed actions | 3 | 47.9 | 51.9 |
| | 4 | 49.5 | 53.0 |
| | 5 | 49.2 | **53.6** |
| | 6 | **49.8** | 53.4 |
| | 7 | 48.6 | 52.3 |

### C.6 REWARD FUNCTION

**Chunk-Level Feedback.** REFLOWTTA uses dense task feedback at the chunk level. At each decision step, the first $T_e$ actions of the predicted action chunk are executed, after which a scalar reward $r_t$ is assigned to the corresponding chunk-level transition in Eq. 4. Thus, adaptation receives feedback throughout deployment rather than relying solely on an episode-level binary success signal.

**Simulation Reward.** Let $d^{\text{g,o}}$ denote the gripper-to-object distance and $d^{o,\text{tar}}$ the object-to-target distance at a step. The two phase-specific distance scores are $\Phi^{\text{reach}} = 1 - \tanh(10 d^{\text{g,o}})$ before the first stable grasp and $\Phi^{\text{place}} = 1 - \tanh(10 d^{o,\text{tar}})$ thereafter. Let $\Phi$ denote the potential used to measure progress across the executed chunk. Writing $\Phi_t^{\text{start}}$ and $\Phi_t^{\text{end}}$ for this potential before and after the executed portion of chunk $t$, the main dense reward component is the chunk's progress:

$$r_{t,\text{prog}}^{\text{sim}} = 3\big(\Phi_t^{\text{end}} - \Phi_t^{\text{start}}\big). \tag{23}$$

The chunk-level feedback used for source-policy learning is based on this progress: moving toward the active target gives positive feedback, whereas moving away gives negative feedback.

**Real-World Reward.** RoboMeter (Liang et al., 2026a), a general-purpose reward model trained on large-scale robot trajectories across tasks and embodiments, is used to estimate task progress from visual observations and the language instruction. Following its eight-frame input setting, the current observation together with the preceding seven observations is used to estimate a normalized task progress $P_t \in [0, 1]$. The reward for an executed chunk is the progress change across its execution:

$$r_t^{\text{real}} = P_{t+1} - P_t. \tag{24}$$

### C.7 BASELINE IMPLEMENTATION DETAILS

**DSRL.** Following the replay-buffer warm-start procedure of DSRL (Wagenmaker et al., 2025), we collect 10 reward-labeled trajectories using the frozen base agent and use them to initialize the replay buffer before adaptation. These base-agent rollouts are excluded from the reported $K$ adaptation episodes; DSRL therefore uses $K + 10$ deployment episodes in total.

**PDF.** PDF (Zang et al., 2026) learns perturbations to autoregressive, discrete action-token logits, whereas our base agent generates actions through flow matching. For a fair comparison, both agent variants start from the same pretrained $\pi_{0.5}$ backbone (Black et al., 2025a) and are trained on the same task-specific data. For PDF, we implement FAST tokenization and autoregressive action decoding to provide the required action-token logits. Both variants use the same evaluation protocol. Before adaptation, their success rates differ by less than 1.3 points on every evaluated task.




Figure 10: **Failure case under an unseen task skill.** Adaptation from pick-and-place manipulation to drawer opening, where the required grasp-and-pull behavior is absent from the training tasks. See § D for details.

**RoboMonkey and AAC.** Following RoboMonkey's reported deployment protocol (Kwok et al., 2025), we use its separately trained VLM action verifier, draw five initial policy samples, and score 16 perturbed candidates at each decision. Following the official AAC protocol (Liang et al., 2026b), we estimate action entropy from 20 sampled action chunks per decision. Unless stated otherwise, other baseline-specific settings follow the protocols of their respective cited works.

### C.8 CODEBASE

Our implementation builds on RLinf (Yu et al., 2026), which provides reusable RL components and standardized interfaces for VLA agents and simulators, including ManiSkill3(Tao et al., 2025), LIBERO(Liu et al., 2023), and LIBERO-Plus(Fei et al., 2026). The source-mean optimizer operates on source coordinates and chunk-level feedback, while simulator-specific adapters handle observations, interactions between the agent's actions and the environment, and reward computation; thus, the same update algorithm can be reused across benchmarks.

## D FAILURE CASES AND SCOPE OF ADAPTATION

REFLOWTTA exploits deployment feedback to reroute the action-generation process toward more favorable trajectories within a frozen VLA. Its effectiveness therefore relies on the required target-domain behaviors remaining accessible under the pretrained generative dynamics. Two challenging regimes further characterize the scope of such adaptation.

**Large Cross-Embodiment Gap.** Large morphological and kinematic differences can exceed the scope of source-space adaptation, as in transfers from larger collaborative manipulators such as Franka Panda to compact arms such as ARX X5 or PiPER. Differences in link geometry, joint configuration, and reachable workspace may require motion strategies that are poorly represented in a policy trained on the source embodiment. Since REFLOWTTA adapts the source distribution while keeping the VLA frozen, its effectiveness depends on suitable target-embodiment behaviors being accessible through the pretrained action-generation process. When such behaviors are absent or insufficiently represented, source rerouting alone may fail to elicit successful execution.

**Task Shift Requiring an Unseen Skill.** A more fundamental boundary arises when deployment requires an action pattern absent from the training tasks. As illustrated in Fig. 10, an agent trained on pick-and-place manipulation is adapted to the instruction *"open the drawer"*. Although adaptation drives the agent toward the drawer, it does not reliably discover the handle or produce the coordinated grasp-and-pull behavior required for opening it. The adapted behavior therefore remains largely limited to approaching the drawer rather than completing the interaction. Unlike object, scene, or moderate embodiment shifts, opening a drawer requires a distinct contact and pulling pattern that was not acquired during training.

Together, these cases indicate that REFLOWTTA is most effective when deployment failures arise from limited access to pretrained capabilities. As the required behavior moves beyond the behavioral support of the frozen VLA, rerouting alone becomes insufficient, marking the boundary between test-time adaptation and the acquisition of genuinely new skills.

## E SUPPLEMENTARY VIDEOS

The archive `Supplementary_Video_41552.zip` contains 11 rollout videos corresponding to Figs. 4, 8, 9, and 10. For compact presentation, the videos are provided at 480p and 2× speed.

# F LIMITATION AND FUTURE WORK

**Limitation.** REFLOWTTA remains bounded by the capabilities already represented in the frozen VLA and cannot readily recover behaviors that lie far beyond its pretrained behavioral support, as further illustrated by the failure cases in Appendix D. Consistently, Tables 2 and 3 show smaller adaptation gains on tasks where the base VLA has little initial competence. In addition, the current formulation accumulates source-space updates within each deployment condition but does not explicitly reuse the learned adaptation across different scenes, tasks, or embodiments, requiring new conditions to be adapted independently. Finally, REFLOWTTA is formulated around the continuous source distribution of diffusion- and flow-based action generators. Extending the same adaptation principle to autoregressive VLAs would require a different interface in their generation process.

**Future Work.** A promising direction is to move beyond condition-specific adaptation toward reusable and structured source-space knowledge. Different tasks or embodiments may correspond to distinct yet partially overlapping favorable regions of the source space; preserving multiple such regions, rather than representing each deployment condition through a single adapted source distribution, could enable previously acquired adaptation experience to be retrieved, composed, and transferred to new conditions. Structured or multimodal source distributions, together with context-conditioned selection of favorable source regions, provide a natural path toward this goal and may substantially reduce the interaction required for recurring or related deployment conditions.

**Social Impact.** REFLOWTTA may facilitate the deployment of pretrained VLA agents by reducing the amount of task-specific data and retraining required for adaptation to new deployment conditions. Rather than fine-tuning the pretrained agent, it keeps the VLA frozen and restricts adaptation to the mean of a low-dimensional source distribution. This constrained form of adaptation preserves the pretrained model parameters and may reduce weight-level behavioral drift that could disrupt previously learned behaviors or induce abrupt and unintended motions during physical deployment. These properties can support more data-efficient and controlled adaptation when transferring pretrained agents across changing scenes, tasks, or embodiments.