

Align-Then-stEer: Adapting the Vision-Language Action Models through Unified Latent Guidance

Yang Zhang^{1,2†}, Chenwei Wang^{1,3†}, Ouyang Lu^{1,4†}, Yuan Zhao¹, Yunfei Ge¹, Zhenglong Sun³, Xiu Li², Chi Zhang¹, Chenjia Bai^{1*}, Xuelong Li^{1*}

¹Institute of Artificial Intelligence, China Telecom, ²Tsinghua University, ³The Chinese University of Hong Kong, Shenzhen, ⁴Northwestern Polytechnical University

†Equal Contributions *Corresponding Authors

Vision-Language-Action (VLA) models pre-trained on large, diverse datasets show remarkable potential for general-purpose robotic manipulation. However, a primary bottleneck remains in adapting these models to downstream tasks, especially when the robot's embodiment or the task itself differs from the pre-training data. This discrepancy leads to a significant mismatch in action distributions, demanding extensive data and compute for effective fine-tuning. To address this challenge, we introduce **Align-Then-stEer (ATE)**, a novel, data-efficient, and plug-and-play adaptation framework. ATE first aligns disparate action spaces by constructing a unified latent space, where a variational autoencoder constrained by reverse KL divergence embeds adaptation actions into modes of the pre-training action latent distribution. Subsequently, it steers the diffusion- or flow-based VLA's generation process during fine-tuning via a guidance mechanism that pushes the model's output distribution towards the target domain. We conduct extensive experiments on cross-embodiment and cross-task manipulation in both simulation and real world. Compared to direct fine-tuning of representative VLAs, our method improves the average multi-task success rate by up to **9.8%** in simulation and achieves a striking **32% success rate gain** in a real-world cross-embodiment setting. Our work presents a general and lightweight solution that greatly enhances the practicality of deploying VLA models to new robotic platforms and tasks.

Date: September 2, 2025

Project: <https://align-then-steer.github.io/>

Code: <https://github.com/TeleHuman/Align-Then-Steer>

Correspondence to: Chenjia Bai (baicj@chinatelecom.cn)



1 Introduction

Vision-Language Action models (VLAs) (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024) have emerged as a powerful solution to empower general-purpose robotic manipulation with broad generalization in tasks and environments. Inspired by the success in scaling data and parameters of Large Language Models (LLMs) and Visual-Language Models (VLMs), the VLA models extend the pre-trained VLM models (Li et al., 2023; Driess et al., 2023; Beyer et al., 2024; Wang et al., 2024) by integrating an action expert module to generate low-level actions, thereby inheriting the common knowledge, semantic reasoning, and instruction-following capabilities of VLMs to facilitate policy learning. Specifically, a representative training recipe for VLA typically consists of two phases: a (i) *pre-training phase*: pretrain the entire VLA model that contains a strong VLM and an action expert on a large corpus containing cross-embodiment datasets (Jang et al., 2021; Collaboration et al., 2023; Khazatsky et al., 2024), which spans diverse robots and tasks to learn a general visuomotor prior at scale, and an (ii) *adaptation phase*: fine-tune the VLA model on a narrowly curated dataset tailored to a particular robot platform and associated tasks, aiming to enhance the manipulation abilities in specific downstream domains.

Though the two-phase paradigm is efficient, it becomes challenging when downstream domains adopt embodiments differing from those in pre-training. For instance, the datasets for pre-training mainly employ various single-arm robots, while the target embodiment might instead utilize dual-arm platform or humanoid robots.

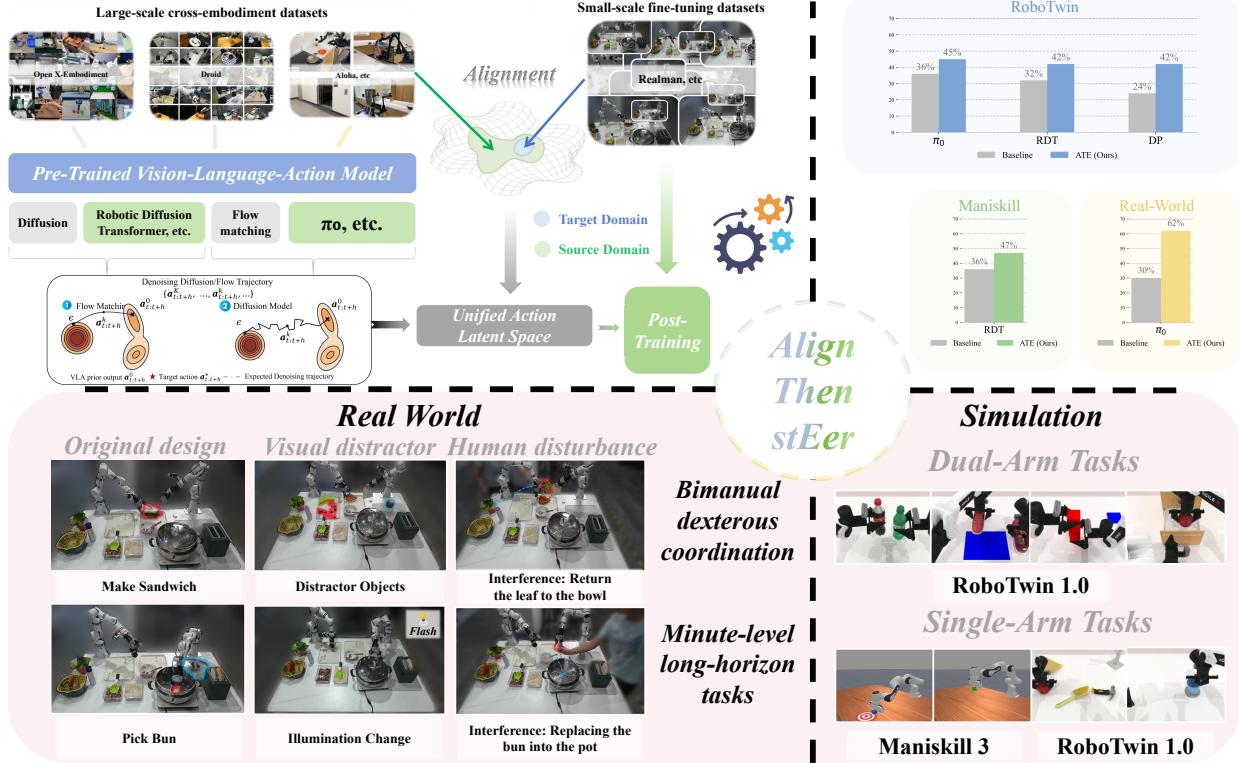


Figure 1 We present **ATE**, a plug-and-play adaptation framework for pre-trained Vision-Language-Action (VLA) models. Unlike prior methods that directly fine-tune VLAs, ATE aligns disparate action spaces into a unified latent representation and steers the VLA’s generation via guidance, enabling data-efficient cross-task and cross-embodiment adaptation. This framework is evaluated in simulation on RoboTwin and ManiSkill benchmarks, as well as on a real-world dual-arm RealMan 7-DoF robot, demonstrating strong generalization, bimanual dexterous coordination, and minute-level long-horizon manipulation, achieving substantial gains in multi-task success rates.

Since VLAs rely on imitation learning that is closely tied to action labels, the discrepancy in adaptation arises as the action spaces and, correspondingly, the action distributions are significantly different from those in the pretraining dataset, often necessitating extensive data collection for the new embodiment. Furthermore, even with the same robot platform, differences in tasks and deployment setup also lead to significant mismatches in the action distributions of the two phases. Recent approaches try to develop adaptation techniques by reducing the training cost at the parameter level (Kim et al., 2024; Zhang et al., 2025b), or by exploring different design decisions (Kim et al., 2025; Pertsch et al., 2025). However, they often struggle to address discrepancies in task distributions and overlook the embodiment mismatches that are significantly more challenging to resolve. As a consequence, efficient adaptation becomes the primary bottleneck for specific robot deployments in target domains. This motivates the central question of this paper: how can we achieve *VLA adaptation for both cross-embodiment and cross-task scenarios under a limited data regime?*

To this end, we propose a data-efficient adaptation method, named **Align-Then-stEer (ATE)**, for cross-embodiment and cross-task VLA adaptation by explicitly mitigating action-distribution mismatch via a two-stage strategy. In the first stage, we construct a unified latent action space to bridge the domain gap between embodiments and tasks involved during pre-training and the adaptation phases. It is achieved by training a standard Variational Autoencoder (VAE) on the pre-training action-labeled dataset to obtain a latent prior, followed by learning a smaller VAE on limited adaptation data with a reverse KL divergence constraint toward the learned pre-trained latent distribution. Thanks to the mode-seeking behavior of reverse KL divergence, this encourages the encoded latent of the target embodiment’s actions to be embedded into a specific mode of the learned prior, yielding a unified and hierarchically structured latent space. In the second stage, inspired by classifier-guided diffusion models (Dhariwal and Nichol, 2021; Bansal et al., 2023), we design a guidance function grounded in the shared latent space that can be seamlessly integrated into the training objective

of diffusion- or flow-based VLAs. This classifier guidance steers the output distribution of the pre-trained VLA toward that of the target embodiment and task, enabling faster and precise adaptation under limited data. Notably, our method requires no modification to the original VLA architecture and introduces only negligible overhead, i.e., adaptation is highly efficiently achieved by training two separate lightweight VAEs. An overview of our method is given in Fig. 1.

Our key contributions are summarized as follows:

- We propose a novel alignment strategy to bridge the action-distribution gap between pre-training and adaptation. By leveraging the mode-seeking behavior of reverse KL, our method constructs a unified, structured latent space that embeds target actions into modes of the pre-training latent distribution.
- We introduce a classifier-guidance mechanism that operates within this unified latent space to explicitly steer the VLA’s output distribution. This enables precise and rapid adaptation to new embodiments and tasks, without requiring additional data.
- The proposed ATE framework is model-agnostic and plug-and-play, seamlessly integrating with various diffusion- or flow-based VLAs without modifying their architecture and with negligible computational overhead.
- We validate our method across different embodiments and tasks in both simulated and real-world manipulation settings. The result demonstrates consistent improvements over baselines that rely on direct supervised fine-tuning, highlighting our approach’s superior adaptation efficiency.

2 Related Works

Vision-Language-Action Models. The long-term objective of robotic manipulation is to develop general-purpose models that can adapt to diverse embodiments and tasks. Vision-Language-Action models (VLAs) is currently a mainstream approach that builds upon the perception and language understanding capabilities of pre-trained VLMs and introduces an additional action prediction module. Leveraging massive datasets of robotic demonstrations for joint pretraining, these models can process language instructions, visual observations, and proprioceptive input to generate continuous action sequences that interact with the physical environment to accomplish complex real-world tasks. Early representative works such as Octo and RT-1 (Brohan et al., 2022; Team et al., 2024) trained Transformer-based policies from scratch on large-scale robotic datasets, while RT-2 and OpenVLA (Zitkovich et al., 2023; Kim et al., 2024) further utilized pre-trained VLMs (e.g., PaLI-X, Prismatic) (Chen et al., 2023a; Anschütz and Le Bras, 2023) and fine-tuned them on robot datasets (e.g., OXE) (Collaboration et al., 2023) via action discretization, achieving improved capabilities by inheriting the knowledge of VLMs. Subsequent methods such as RDT-1B, H-RDT, DexVLA, DiVLA, DexGraspVLA, Dita, and etc. (Liu et al., 2024; Bi et al., 2025; Wen et al., 2025a,c; Zhong et al., 2025; Hou et al., 2025) adopt diffusion-based architectures to represent the multimodality in robot data. Recently, a new generation of VLA models—such as π_0 , GROOT, $\pi_{0.5}$, SmolVLA, and TinyVLA (Black et al., 2024; NVIDIA, 2025; Intelligence et al., 2025; Shukor et al., 2025; Wen et al., 2025b) adopt a dual system compositional architecture that incorporates a generative action expert based on diffusion or flow matching (Liu et al., 2023a; Lipman et al., 2023) to produce continuous action sequences, significantly improving the prediction accuracy and physical executability. However, they mostly focus on architectures and objectives in the pre-training phase, while we aim to improve the efficiency for cross-embodiment and cross-task adaptation on pretrained VLAs.

Policy Adaptation of VLAs. Despite the promise of VLAs for general-purpose robot manipulation, current approaches still suffer from high fine-tuning costs for embodiment and task mismatches. Several approaches construct latent action representation for different data sources (Ye et al., 2025; Bu et al., 2025; NVIDIA, 2025), perform Kinematics retargeting or eigengrasps (Bauer et al., 2025; Yuan et al., 2025), or manually design a physically interpretable action space for different robot platforms (Liu et al., 2024; Zheng et al., 2025), while they cannot easily adapt to different robot platforms since the target action distribution can be significantly different from the original one. RoboOS adopts advanced architectures by building skill library and shared action space for different embodiments, while is require complex module construction and coupling. Other works improve the adaptation efficiency of VLAs by caching visual tokens and selectively updating task-relevant ones (Xu et al., 2025), compressing high-frequency action sequences into low-frequency tokens

via discrete cosine transform (Pertsch et al., 2025), adopting LoRA-based update (Wen et al., 2025b), or performing dynamic layer activation for distillation (Zhang et al., 2025a). However, these methods are often restricted to auto-regressive manner and overlook the diffusion and flow-matching architectures. Meanwhile, solely increasing the training efficiency is still limited in cross-embodiment adaptation as the domain gap would be hard to bridge via direct tuning. In contrast, we take a novel perspective by using mode seeking property of VAEs to bridge the gap of action distributions in pre-train and adaptation, and also employ latent guidance that are seamless compatible for diffusion and flow-matching VLAs. More discussion on the recent works of constructing unified action spaces to enhance the ability of VLAs are provided in the Appendix A.

3 Preliminaries

Training objective of VLAs. The training of VLA models typically contains two stages: (i) obtain an initial policy via pre-training the model on a cross-embodiment action-labeled dataset $\mathcal{D}_{\text{pretrain}}$, where we denote the set of embodiments included in it as $\mathcal{M}_{\text{pretrain}} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n\}$; (ii) finetune the policy on a narrow but high-quality dataset $\mathcal{D}_{\text{adaptation}}$ collected from the target embodiment $\mathcal{E}_{\text{target}}$ for downstream applications. Formally, the optimization objective of the VLA policy π_θ is to maximize the log-likelihood of an action chunk $\mathbf{a}_{t:t+h}$ given an observation \mathbf{o}_t and a language-format task instruction l sampled from the dataset, which can be formulated as follows:

$$\max_{\theta} \mathbb{E}_{(\mathbf{a}_{t:t+h}, \mathbf{o}_t, l) \sim \mathcal{D}} [\log \pi_\theta(\mathbf{a}_{t:t+h} | \mathbf{o}_t, l)], \quad (1)$$

where \mathbf{o}_t contain single-view or multi-view RGB observations $\mathbf{I}_t^1, \mathbf{I}_t^2, \dots, \mathbf{I}_t^n$, and the proprioceptive state \mathbf{q}_t usually describes the joint position and angle of the robot.

Diffusion Models and Flow Matching. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) are generative models where data generation is modeled as a denoising process. Consider a forward diffusion process that gradually adds Gaussian noise to samples $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ over T steps and produces a sequence of noisy samples $\mathbf{x}_1, \dots, \mathbf{x}_T$ that satisfies $q(\mathbf{x}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})$ and $q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t)I)$, where $\{\alpha_t \in (0, 1)\}_{t=1}^T$ is a predefined noise scheduler. The denoising diffusion model reverses the above process to approximate the data distribution $p_\theta(\mathbf{x}_0)$ which is parameterized as $p_\theta(\mathbf{x}_0) = \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} = \int p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) d\mathbf{x}_{1:T}$. By minimizing the following denoising loss function introduced by Ho et al. (2020),

$$\mathcal{L}(\theta) = \mathbb{E}_{t, \mathbf{x}_0, \epsilon} [\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2], \text{ where } \bar{\alpha}_t = \prod_{i=1}^t \alpha_i, \quad (2)$$

we can get a noise prediction network $\epsilon_\theta(\mathbf{x}_t, t)$ to perform the reverse diffusion process - it begins by sampling $\mathbf{x}_T \sim \mathcal{N}(0, I)$, and then iteratively applies the update

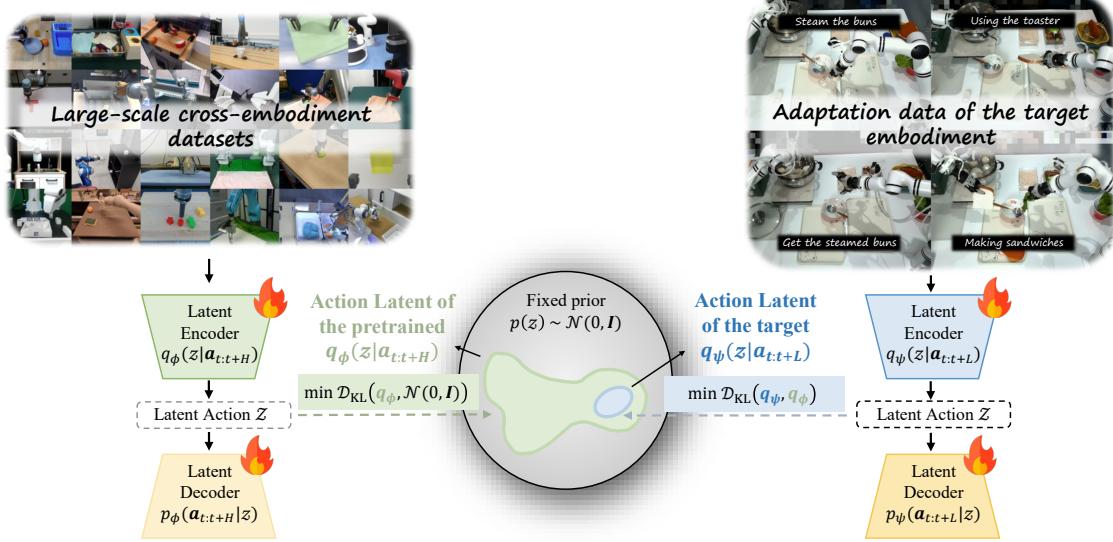
$$\mathbf{x}_{k-1} = \frac{1}{\sqrt{\alpha_k}} (\mathbf{x}_k - \sqrt{1 - \bar{\alpha}_k} \epsilon_\theta(\mathbf{x}_k, k)) + \sigma_k \epsilon_k, \quad (3)$$

where $\epsilon_k \sim \mathcal{N}(0, I)$, and σ_k is a function of k and depends on the noise scheduler.

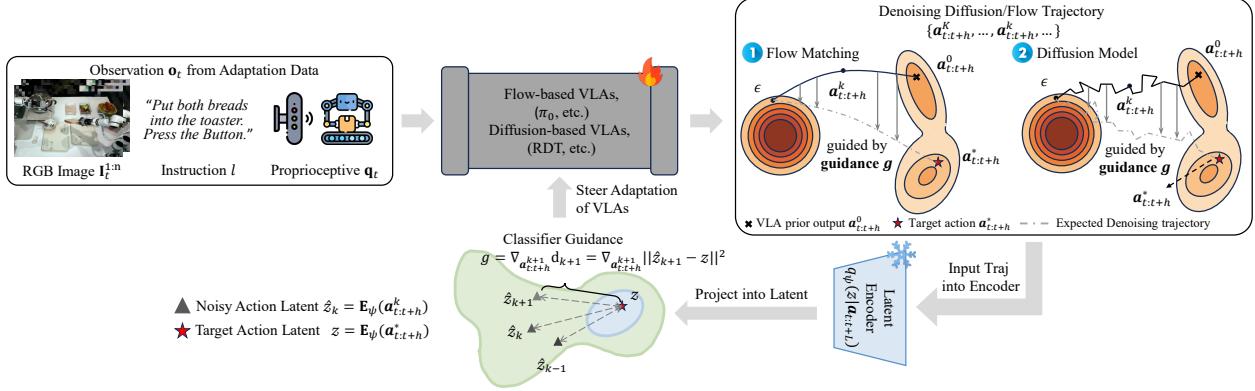
Beyond denoising diffusion, flow matching is another generative modeling framework that implements data generation as a probability path from a known source distribution to the data target distribution (Lipman et al., 2023; Liu et al., 2023b; Lipman et al., 2024). Assuming the source sample and target sample are denoted as \mathbf{x}_1 and \mathbf{x}_0 respectively, and flow matching timesteps are denoted as $\tau \in [0, 1]$. By instantiating the path as a linear-Gaussian path given by $\mathbf{x}_\tau \sim \mathcal{N}(\tau \mathbf{x}_0, (1 - \tau)I)$, flow matching operates by first sampling noise from a standard Gaussian and then processes this noise through a deterministic process to produce a sample from the target distribution, similar to the denoising process. The flow v_θ that maintains the probability path is typically trained via optimizing a simple conditional flow matching loss function,

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau, \mathbf{x}_0, \epsilon} [\|v_\theta(\tau \mathbf{x}_0 + (1 - \tau)\epsilon, \tau) - (\mathbf{x}_0 - \epsilon)\|^2]. \quad (4)$$

In robotic manipulation, there can be dozens of possible actions \mathbf{a}_t to accomplish the task given the same language instruction l and observation. Motivated by the capability of diffusion models and flow matching on



(a) Stage 1: Learning the Unified Action Latent Space



(b) Stage 2: Steering Efficient Adaptation with Latent Guidance

Figure 2 The overview of ATE framework. (a) In the first stage, we construct a unified action space to bridge the embodiment gap in pretraining and adaptation stages by utilizing the mode-seeking behavior of asymmetric VAEs. (b) In the second stage, we integrate classifier guidance in diffusion and flow-based VLAs to steer the pretrained policy towards the target action distribution with specific robot platforms.

providing high-precision continuous representation and modeling multi-modal distribution, recent VLAs (Black et al., 2024; Liu et al., 2024; Intelligence et al., 2025; Bjorck et al., 2025) have incorporated diffusion-based or flow-based action prediction to boost performance.

Problem Setting. In this work, we assume access to a pre-trained diffusion-based or flow-based policy, which can be either a VLA model or Vision-Action model like Diffusion Policy (Chi et al., 2023). Our goal is to adapt the pretrained model in $\mathcal{D}_{\text{pretrain}}$ to a specific embodiment $\mathcal{E}_{\text{target}}$ given an adaptation dataset $\mathcal{D}_{\text{adaptation}}$, where $\mathcal{E}_{\text{target}}$ is not involved in pre-training, i.e., $\mathcal{E}_{\text{target}} \notin \mathcal{M}_{\text{pretrain}}$. In addition, the tasks contained in $\mathcal{D}_{\text{adaptation}}$ can also be significantly different from that in pre-training. For clarity, we use the superscript k to denote the diffusion or flow-matching timestep, and the subscript t to denote the trajectory timestep.

4 Methodology

The proposed ATE algorithm mainly consists of two stages: (i) we extract a unified action latent space to compactly encompass the actions of different embodiments, thus bridging the domain gap between the embodiments in the pre-training and adaptation phase (details in Section 4.1); (ii) then, we design a guidance

function grounded in the unified latent space and leverage the resulting classifier guidance to explicitly steer the fine-tuning towards our desired data distribution during adaptation (details in Section 4.2). Fig. 2 provides an overview of our entire procedure.

4.1 Alignment through Unified Action Latent Space

A pre-trained VLA acquires a general visuomotor prior by modeling the distribution $p(\bar{\mathbf{a}}_{t:t+H-1}|\mathbf{o}_t, l)$, where the action chunk $\bar{\mathbf{a}}_{t:t+H-1}$ of length H , observation \mathbf{o}_t and language instruction l are sampled from the pre-training data $\mathcal{D}_{\text{pretrain}}$. Such a prior effectively captures the action distribution $p_{\mathcal{D}_{\text{pretrain}}}(\bar{\mathbf{a}}_{t:t+H-1})$ from the pre-training embodiments $\mathcal{M}_{\text{pretrain}}$. However, due to the inherent differences in degrees of freedom (DoFs) and physical representation (e.g., joint angles, end-effector poses, and joint torques), the raw action-labeled data from the target embodiment $\mathcal{E}_{\text{target}}$ can vary significantly from the action generated by the pre-trained VLA. This leads to a substantial **domain gap** that hinders efficient adaptation. To bridge this gap, we employ two distinct Variational Autoencoders (VAEs) to encode the fundamentally disparate pre-training and adaptation action data into a single, unified, and compact action latent space.

Learning Pre-training Action Latent. We construct a pre-training action-VAE $\mathcal{V}_{\text{pretrain}} := \{\mathbf{E}_\phi, \mathbf{D}_\phi\}$ to compress all possible action chunks in the pre-training data into a low-dimensional latent space \mathcal{Z} , which consists of a Transformer encoder \mathbf{E}_ϕ and a Transformer decoder \mathbf{D}_ϕ . The encoder takes an action chunk $\bar{\mathbf{a}}_{t:t+H-1}$ of length H as input, and outputs the parameters of a Gaussian distribution $q_\phi(z|\bar{\mathbf{a}}_{t:t+H-1})$ to represent the action latent. Then we use the reparameterization trick to sample a latent vector $z \in \mathbb{R}^d$. Inspired by prior work on motion sequence VAEs (Petrovich et al., 2021; Chen et al., 2023b), we similarly prepend two learnable tokens, μ_{token} and Σ_{token} , to the input. The corresponding outputs from the encoder are then used to convert fixed-length action chunks into a single, chunk-level latent representation. The decoder, built upon a Transformer decoder with a cross-attention mechanism, takes H zero embeddings as a query and the latent vector z as key and value, and generates a reconstructed action chunk of length H .

We instantiate $\mathcal{V}_{\text{pretrain}}$ as an InfoVAE (Zhao et al., 2017) rather than a vanilla VAE (Kingma and Welling, 2013). Formally, it is optimized by maximizing the log-likelihood term together with a KL regularization term towards the fixed latent prior distribution and a mutual information maximization term that encourages high mutual information between $\bar{\mathbf{a}}_{t:t+H-1}$ and z . The objective can be simplified as,

$$\mathcal{L}(\phi; \mathcal{D} \leftarrow \mathcal{D}_{\text{pretrain}}) = \mathbb{E}_{p_{\mathcal{D}}(\bar{\mathbf{a}}_{t:t+H-1})} \mathbb{E}_{q_\phi(z|\bar{\mathbf{a}}_{t:t+H-1})} [\log p_\phi(\bar{\mathbf{a}}_{t:t+H-1}|z)] - (1 - \alpha) \mathbb{E}_{p_{\mathcal{D}}(\bar{\mathbf{a}}_{t:t+H-1})} [D_{\text{KL}}(q_\phi(z|\bar{\mathbf{a}}_{t:t+H-1}) \| p(z))] - (\alpha - \lambda - 1) D_{\text{KL}}(q_\phi(z) \| p(z)), \quad (5)$$

where $\mathcal{V}_{\text{pretrain}}$ is parameterized by ϕ , $p_{\mathcal{D}}(\bar{\mathbf{a}}_{t:t+H-1})$ is an abbreviation for $p_{\mathcal{D}_{\text{pretrain}}}(\bar{\mathbf{a}}_{t:t+H-1})$, and α and λ are hyperparameters that control penalty strength. The latent prior $p(z)$ is set as a unit Gaussian $\mathcal{N}(0, I)$.

Embedding Adaptation Action Latent into Pre-training Action Latent Space. With the pre-training action VAE $\mathcal{V}_{\text{pretrain}}$, we can obtain the learned action latent distribution $q_\phi(z)$ by marginalizing over all action chunks in $\mathcal{D}_{\text{pretrain}}$. To create a unified action latent space, we train a separate adaptation action VAE $\mathcal{V}_{\text{adaptation}} := \{\mathbf{E}_\psi, \mathbf{D}_\psi\}$ which is parameterized as ψ , with a KL regularization towards the learned pre-training action latent distribution. Denoting the action chunk of length L in the adaptation data as $\tilde{\mathbf{a}}_{t:t+L-1}$, the learning objective of $\mathcal{V}_{\text{adaptation}}$ is written as follows,

$$\mathcal{L}(\psi; \mathcal{D} \leftarrow \mathcal{D}_{\text{adaptation}}) = \mathbb{E}_{p_{\mathcal{D}}(\tilde{\mathbf{a}}_{t:t+L-1})} \mathbb{E}_{q_\psi(z|\tilde{\mathbf{a}}_{t:t+L-1})} [\log p_\psi(\tilde{\mathbf{a}}_{t:t+L-1}|z)] - (1 - \alpha) \mathbb{E}_{p_{\mathcal{D}}(\tilde{\mathbf{a}}_{t:t+H-1})} [D_{\text{KL}}(q_\psi(z|\tilde{\mathbf{a}}_{t:t+L-1}) \| q_\phi(z))] - (\alpha - \lambda - 1) D_{\text{KL}}(q_\psi(z) \| q_\phi(z)), \quad (6)$$

where $p_{\mathcal{D}}(\tilde{\mathbf{a}}_{t:t+L-1})$ is an abbreviation for $p_{\mathcal{D}}(\tilde{\mathbf{a}}_{t:t+L-1}) p_{\mathcal{D}_{\text{adaptation}}}(\tilde{\mathbf{a}}_{t:t+L-1})$. The adaptation action VAE has the same architecture and variant as the pre-training VAE. Minimizing the reverse KL divergence $D_{\text{KL}}(q_\psi(z|\tilde{\mathbf{a}}_{t:t+L-1}) \| q_\phi(z))$ instead of $D_{\text{KL}}(q_\phi \| q_\psi)$ ensures that the learned adaptation action latent distribution is mode-seeking (Bishop and Nasrabadi, 2006). Therefore, this effectively embeds the adaptation latent distribution $q_\psi(z)$ into one mode of the pre-trained action latent distribution $q_\phi(z)$, yielding a unified and compact latent space \mathcal{Z} .

Implementation Details. The purpose of using two VAEs is to bridge the distribution gap of the output action chunks between the pre-trained VLA and the adaptation data. Thus, the pre-training VAE $\mathcal{V}_{\text{pretrain}}$ uses

the same action chunk length H as in the pre-training phase, while the adaptation action VAE $\mathcal{V}_{\text{adaptation}}$ adopts the action chunk length L used in the adaptation phase. Following the guidelines of Zhao et al. (2017), we replace the last KL term in both Eq. (5) and Eq. (6) with Maximum-Mean Discrepancy (Li et al., 2015; Dziugaite et al., 2015) for more tractable optimization. The learned pre-training action latent distribution $q_\phi(z)$ in Eq. (6) is approximated as $\mathcal{N}(\mu_\phi, \Sigma_\phi)$, where μ_ϕ and Σ_ϕ are calculated over all sampled pre-training action latents. A detailed algorithm for training both VAEs is provided in Alg. 1.

4.2 Classifier Guidance for Steering Adaptation

To facilitate efficient adaptation of a pre-trained diffusion or flow-based VLA, we introduce the classifier guidance (Dhariwal and Nichol, 2021) to explicitly steer the fine-tuning process. The core idea is to design a guidance function that measures the discrepancy between the generated actions and the target action distribution within the unified latent space \mathcal{Z} , and then use the gradient of this function to guide the policy update. This approach enables highly efficient adaptation without requiring additional data budget. In the following, we first introduce the formulation of classifier guidance in diffusion and flow-based VLAs; then we detail the guidance function according to the learned latent action space.

Classifier Guidance in Diffusion-based VLAs. Note that the action latent z is only determined by the action chunk $\mathbf{a}_{t:t+h}$, i.e., independent of the observation \mathbf{o}_t and language instruction l . As the conditioning label we use for guidance is based solely on the latent z , we thus omit these conditions and treat it as an unconditional diffusion model $\pi_\theta(\cdot) := p_\theta(\cdot)$ for notational simplicity. A diffusion model typically predicts the noise $\epsilon_\theta(\mathbf{a}_{t:t+h}^k, k)$ that corrupts an initial action chunk $\mathbf{a}_{t:t+h}^0$ to a noisy version $\mathbf{a}_{t:t+h}^k$. According to Luo (2022), by Tweedie's Formula, it can be used to derive a score function,

$$\nabla_{\mathbf{a}_{t:t+h}^k} \log p_\theta(\mathbf{a}_{t:t+h}^k) = -\frac{1}{\sqrt{1-\bar{\alpha}_k}} \epsilon_\theta(\mathbf{a}_{t:t+h}^k, k). \quad (7)$$

To generate a desired action chunk conditioned on a label y , we have to sample from the conditional distribution $p_\theta(\mathbf{a}_{t:t+h}^k|y)$. Using Bayes' rule, the score function for this conditional distribution is:

$$\nabla_{\mathbf{a}_{t:t+h}^k} \log p_\theta(\mathbf{a}_{t:t+h}^k|y) = \nabla_{\mathbf{a}_{t:t+h}^k} \log p_\theta(\mathbf{a}_{t:t+h}^k) + \nabla_{\mathbf{a}_{t:t+h}^k} \log p(y|\mathbf{a}_{t:t+h}^k).$$

By substituting the score function in Eq. (7), we obtain a calibrated noise prediction $\hat{\epsilon}$ that incorporates the guidance gradient $g = \nabla_{\mathbf{a}_{t:t+h}^k} \log p(y|\mathbf{a}_{t:t+h}^k)$, which corresponds to the score of the joint distribution of the action chunk and the conditioning label, as

$$\hat{\epsilon}(\mathbf{a}_{t:t+h}^k, k) := \epsilon_\theta(\mathbf{a}_{t:t+h}^k, k) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{a}_{t:t+h}^k} \log p(y|\mathbf{a}_{t:t+h}^k) = \epsilon_\theta(\mathbf{a}_{t:t+h}^k, k) - \sqrt{1-\bar{\alpha}_t} g, \quad (8)$$

where we delay the discussion of the exact expression of g subsequently. By incorporating this modified noise prediction into the original denoising objective, we explicitly steer the fine-tuning stage of the VLA. The final objective is reformulated as,

$$\mathcal{L}(\theta) = \mathbb{E}_{k, \epsilon, (\mathbf{o}_t, \mathbf{a}_{t:t+h}^0, l) \sim \mathcal{D}_{\text{adaptation}}} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k} \mathbf{a}_{t:t+h}^0 + \sqrt{1-\bar{\alpha}_k} \epsilon, k, \mathbf{o}_t, l) + \sqrt{1-\bar{\alpha}_k} \cdot \lambda \cdot g \right\|^2 \right], \quad (9)$$

where λ is a guidance scale that modulates the influence of the guidance.

Classifier Guidance in Flow-based VLAs. Similarly, for those VLAs that model the generation of action chunks with Flow Matching framework, we also provide an updated objective that directly incorporates the classifier guidance into the learning of the flow. As proven by Lipman et al. (2024), supposing a velocity field v_θ is an instance of linear Gaussian flows $\mathbf{x}^\tau = \tau \mathbf{x}^1 + (1-\tau) \mathbf{x}^0$ where $q(\mathbf{x}^1)$ is the target data distribution and $p(\mathbf{x}^0) = \mathcal{N}(0, I)$ is the source distribution, the transformation between marginal velocity fields and score function for unconditional distributions satisfies

$$v_\theta(\mathbf{x}^\tau) = \frac{1}{\tau} \mathbf{x}^\tau + \frac{1-\tau}{\tau} \nabla_{\mathbf{x}^\tau} \log p_\theta(\mathbf{x}^\tau). \quad (10)$$

We refer readers to [Lipman et al. \(2024\)](#) for a detailed proof. Similarly, when we condition the velocity field $v_\theta(\mathbf{a}_{t:t+h}^\tau|y)$ on a label y , we have

$$\begin{aligned}\hat{v}_\theta(\mathbf{a}_{t:t+h}^\tau|y) &= \frac{1}{\tau}\mathbf{a}_{t:t+h}^\tau + \frac{1-\tau}{\tau}\nabla_{\mathbf{a}_{t:t+h}^\tau} \log p_\theta(\mathbf{a}_{t:t+h}^\tau|y) \\ &= \frac{1}{\tau}\mathbf{a}_{t:t+h}^\tau + \frac{1-\tau}{\tau} \left[\nabla_{\mathbf{a}_{t:t+h}^\tau} \log p_\theta(\mathbf{a}_{t:t+h}^\tau) + \nabla_{\mathbf{a}_{t:t+h}^\tau} \log p(y|\mathbf{a}_{t:t+h}^\tau) \right] \\ &= v_\theta(\mathbf{a}_{t:t+h}^\tau) + \frac{1-\tau}{\tau}\nabla_{\mathbf{a}_{t:t+h}^\tau} \log p(y|\mathbf{a}_{t:t+h}^\tau).\end{aligned}\quad (11)$$

The final objective for fine-tuning the pre-trained velocity field v_θ is thus rewritten as,

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau, \epsilon, (\mathbf{o}_t, \mathbf{a}_{t:t+h}^0, l) \sim \mathcal{D}_{\text{adaptation}}} \left[\left\| v_\theta(\tau\mathbf{a}_{t:t+h}^0 + (1-\tau)\epsilon, \tau, \mathbf{o}_t, l) + \frac{1-\tau}{\tau} \cdot \lambda \cdot g - (\mathbf{a}_{t:t+h}^0 - \epsilon) \right\|^2 \right], \quad (12)$$

where λ is the guidance scale.

Designing a Guidance Function for Efficient Adaptation. The goal of fine-tuning a pre-trained VLA is to fit the action distribution of the adaptation data. Despite the inherent heterogeneity of actions from different embodiments, we can effectively measure a valid discrepancy between them by leveraging the unified and compact action latent space \mathcal{Z} , established in Section 4.1. To facilitate efficient adaptation, we design a guidance function that measures how closely an intermediate action chunk during the denoising process, $\hat{\mathbf{a}}_{t:t+h}^k$, aligns with the ground truth action chunk, $\mathbf{a}_{t:t+h}^0$, from the adaptation data. Inspired by [Carvalho et al. \(2023\)](#); [Liang et al. \(2025\)](#), we formulate the classifier induced by such a guidance function as an energy-based model,

$$p_\psi(y|\hat{\mathbf{a}}_{t:t+h}^k) = \frac{1}{Z_\psi} \exp(-\|\mathbf{E}_\psi(\hat{\mathbf{a}}_{t:t+h}^k) - \mathbf{E}_\psi(\mathbf{a}_{t:t+h}^0)\|^2),$$

where \mathbf{E}_ψ is the encoder of the adaptation action VAE and Z_ψ is a normalizing constant dependent on ψ . Then the classifier guidance g can be easily calculated via,

$$g = \nabla_{\hat{\mathbf{a}}_{t:t+h}^k} \log p_\psi(y|\hat{\mathbf{a}}_{t:t+h}^k) \propto -\nabla_{\hat{\mathbf{a}}_{t:t+h}^k} \|\mathbf{E}_\psi(\hat{\mathbf{a}}_{t:t+h}^k) - \mathbf{E}_\psi(\mathbf{a}_{t:t+h}^0)\|^2, \quad (13)$$

where we use \mathbf{E}_ψ to encode both noisy actions and clean actions. One may ask whether the encoder \mathbf{E}_ψ can robustly encode noisy action chunks. As identified by [Zhao et al. \(2017\)](#), the vanilla VAE ([Kingma and Welling, 2013](#)) suffers from issues such as posterior collapse and inaccurate amortized latent inference, while the InfoVAE effectively addresses these limitations, making it an appropriate alternative for our deliberately designed guidance function. Since the guidance relies on the ground truth action chunk, which is not available during VLA inference, we incorporate this guidance explicitly into the training objective, as shown in Eqs. (9) and (12). During training, we use the noisy action chunk $\hat{\mathbf{a}}_{t:t+h}^k$ corrupted from its ground truth $\mathbf{a}_{t:t+h}^0$ as a reliable substitute for the corresponding intermediate action chunk during the denoising process.

This gradient guidance g allows us to explicitly calibrate the transitions at any noise level k , pushing the VLA's output consistently towards a latent representation that reduces the distance to the target distribution throughout the whole denoising process. As shown in Fig. 2, our method not only steers the pre-trained VLA towards the adaptation data, but also constrains its output to remain within the unified action latent space which happens to be a specific mode of the pre-training action latent space. By preventing the VLA from deviating far from this structured latent manifold, our method also implicitly helps preserve valuable visuomotor prior knowledge acquired during pre-training. Detailed algorithmic descriptions for how we steer the adaptation of diffusion-based/flow-based VLAs are provided in Algs. 2 and 3.

5 Experiments

To demonstrate the efficiency of ATE in facilitating the adaptation process across embodiments and tasks, we evaluate our approach in both simulated and real-world settings. Through extensive experiments, our aim is to answer the following key questions: **Q1:** How efficiently does ATE facilitate the VLA's adaptation

under cross-task and cross-embodiment regimes? (Section 5.2) **Q2**: To what extent does ATE enhance the generalization of the fine-tuned VLA against diverse perturbations, such as variations in object placement, lighting conditions, and visual distractors? (Section 5.3) **Q3**: How critical is our proposed aligned and structured action latent space to achieving efficient cross-embodiment and cross-task adaptation? (Section 5.4)

5.1 Experimental Setup

To demonstrate the efficiency of ATE in facilitating the adaptation process across embodiments and tasks, we evaluate our approach in both simulated and real-world settings.

Experimental Environment. For simulation, we conduct experiments on two recently released benchmarks: **RoboTwin 1.0** (Mu et al., 2025) and **ManiSkill3** (Tao et al., 2024). RoboTwin 1.0 provides 17 diverse single- and dual-arm manipulation tasks (e.g., tool adjustment, dual-bottle picking) that are largely absent from mainstream VLA pretraining datasets (Open X-Embodiment, Droid, and etc.). ManiSkill3 instead focuses on contact-rich single-arm manipulation (e.g., cube pushing, cube picking), emphasizing precise low-level control. Together, these benchmarks allow us to evaluate ATE under both bimanual and contact-rich regimes, directly addressing **Q1** regarding cross-task efficient adaptation.

For real-world experiments, we deploy policies trained using ATE on a dual-arm **RealMan** robot, where each arm has 7 DoF—distinct from the 6-DoF single-arm embodiments predominantly used in mainstream VLA pretraining datasets (e.g., RDT, π_0). We design **four long-horizon dual-arm tasks** requiring precise insertion and tight collaboration, along with an additional **tool-use task** (details are provided in Appendix B). These tasks are particularly challenging for current VLA models, as they require bimanual coordination, tool interaction, and long-horizon reasoning capabilities that remain challenging for existing VLAs. This setting provides a strong testbed to evaluate how efficiently ATE facilitates adaptation across both task and embodiment shifts (**Q1**) and to probe robustness under real-world perturbations such as object placement, lighting, and visual distractors (**Q2**). We refer the detail of real-robot setup in Appendix C.

Baselines. Since ATE is designed to be plug-and-play, it can be directly applied to existing diffusion- and flow matching-based architectures. We benchmark it on three representative models. For diffusion-based models, we include the **Diffusion Policy (DP)**, a foundational diffusion architecture; and **RDT-1B** (Liu et al., 2024), a diffusion-based VLA tailored for bimanual manipulation that leverages a scalable transformer and a physically interpretable unified action space for transferable across robot embodiments. For flow matching-based models, we include π_0 ¹ (Black et al., 2024), a generalist policy built on pre-trained vision-language models, supporting multi-task and multi-embodiment manipulation with strong instruction-following and efficient adaptation ability. Together, these baselines represent the leading paradigms of robotic foundation models, allowing us to evaluate how our aligned and structured action latent space contributes to efficient adaptation (**Q3**). **Note that in all cases, the baseline comparison is against direct fine-tuning of these architectures without our method.**

Implementation. Our implementation consists of two main components.

Firstly, we adopt a two-step Info-VAE training scheme to learn structured latent representations over action chunks, which facilitates efficient guidance during downstream policy adaptation. With only robot action data required, the action-latent learning is lightweight while capturing informative latent structure. In step-1, the training of VAE is performed on large-scale VLA pretraining datasets, aiming to capture transferable latent structure across diverse robot platforms and tasks. The model uses a latent dimension of 512 and is trained with a mutual information term to encourage informative latent variables and alignment between pretraining and downstream action spaces. The training of step-1 typically takes around 12 hours. Then, step-2 fine-tuning is conducted on small sets of domain-specific data, further aligning the action distribution of the target domain to the latent space. This step is extremely fast, taking less than 0.5 hours.

- **Step 1 (General latent learning):** We perform general representation learning on large-scale datasets, which is used in the pretraining stage of VLAs. The datasets for training VAEs in this stage include DROID, Kuka, ALOHA, and selected subsets of Open X-Embodiment, aiming to capture transferable latent structure across diverse robot platforms and tasks.

¹Here we adopt the implementation of π_0 from Huggingface Lerobot Community (<https://github.com/huggingface/lerobot>) across all experiments.

- **Step 2 (Domain-specific latent tuning):** We finetune the pre-trained VAEs with only a small set of data from the target domains (i.e., RoboTwin-1.0, ManiSkill3, and Real-robot setup) that have specific embodiments and tasks. Specifically, we use 100 trajectories per task for RDT-based models, and 50 trajectories for π_0 - and DP-based models.

Table 1 Info-VAE training configuration across stages and policy types.

Policy	Dataset Source	Episodes	Epochs
Step 1			
DP	RoboTwin 1.0: 5 tasks	50/task	1000
RDT	Open X-Embodiment (subset), DROID, Kuka, ALOHA	3000	1000
π_0	Open X-Embodiment (subset), DROID	3000	1000
Step 2			
DP	RoboTwin 1.0: 1 task	50/task	200
RDT	RoboTwin 1.0: 17 tasks, ManiSkill3: 2 tasks	100/task	200
π_0	RoboTwin 1.0: 17 tasks, Real-World: 4 long-horizon tasks	50/task	200

Secondly, the pretrained Info-VAE is used to provide structured latent information for a lightweight guidance module, which facilitates efficient adaptation of diffusion- or flow-matching-based VLAs. During policy fine-tuning, the guidance module leverages the Info-VAE latent space to measure the discrepancy between predicted and target actions and applies gradient signals to steer the policy updates toward the target distribution. The Info-VAE parameters are frozen, with guidance enhanced via random latent perturbations and dynamically scaled according to the current generation step. A tunable parameter controls the overall strength of the guidance, allowing a flexible trade-off between adaptation efficiency and stability. This approach enables rapid adaptation without requiring additional data, and integration into existing VLA training pipelines is straightforward and computationally lightweight. Full algorithmic details, including the latent-space guidance formulation, are provided in the Appendix E.

5.2 Main Results

To answer **Q1**, we first evaluate the proposed ATE method in simulation using two representative benchmarks: RoboTwin 1.0 and ManiSkill3. Task success rate is used as the performance metric to compare our method with baseline models, including RDT, π_0 , and DP. The results demonstrate that our method consistently enhances performance across tasks and architectures, particularly under limited data conditions.

Simulation Results on RoboTwin 1.0. We evaluate both RDT and π_0 on 17 manipulation tasks from the RoboTwin 1.0 benchmark, with results summarized in Table 2. Our ATE variants significantly outperform the original models, achieving average success rate improvements of +10% for RDT and +9% for π_0 , with gains exceeding +30% on challenging tasks such as *put apple cabinet* and *empty cup place*. In addition to higher final performance, our method demonstrates improved sample efficiency, surpassing the RDT baseline at just 70k steps compared to 90k. Fig. 3 illustrates the performance of both the DP baseline and the ATE variant in in-distribution and out-of-distribution settings. Overall, our method consistently improves success rates and accelerates convergence, with particularly large gains on tasks where the baseline struggles. Examining the training curves at 100, 200, and 300 epochs shows that our method reaches higher success rates earlier and maintains strong performance on the most challenging tasks. For out-of-distribution settings such as *dual bottles pick easy* and *shoe place*, the ATE variant consistently surpasses the baseline, with the success rate in *shoe place* nearly doubling at 100 epochs. Similar trends are observed in in-distribution settings, such as *put apple cabinet* and *empty cup place*. The performance gain can be attributed to two key advantages of our algorithm: (i) the unified action latent space, which compactly represents actions of different embodiments and tasks, and bridges the domain gap between pre-training and adaptation; and (ii) the latent space guidance, which explicitly steers fine-tuning towards the target action distribution, which ensures efficient adaptation while preserving valuable visuomotor priors, resulting in both faster convergence and higher final success rates.

Simulation Results on ManiSkill3. The effectiveness of the proposed approach is further validated on the ManiSkill3 benchmark using RDT as the policy backbone. This simulator emphasizes contact-rich manipulation and offers a diverse suite of single-arm tasks. As shown in Table 3, the success rates on two representative

Table 2 The comparison of task success rates on RoboTwin 1.0 benchmark

Task	Method			
	RDT-1B (Liu et al., 2024)	RDT-1B + ATE (Ours)	π_0 (Black et al., 2024)	$\pi_0 + \text{ATE}$ (Ours)
Block Hammer Beat	52%	71% (↑ 19)	38%	44% (↑ 6)
Block Handover	69%	91% (↑ 22)	80%	92% (↑ 12)
Blocks Stack (Easy)	10%	31% (↑ 21)	30%	50% (↑ 20)
Blocks Stack (Hard)	1%	7% (↑ 6)	8%	7% (↓ 1)
Bottle Adjust	53%	37% (↓ 16)	39%	45% (↑ 6)
Container Place	34%	55% (↑ 21)	56%	59% (↑ 3)
Diverse Bottles Pick	18%	24% (↑ 6)	20%	40% (↑ 20)
Dual Bottles Pick (Easy)	76%	87% (↑ 11)	48%	85% (↑ 37)
Dual Bottles Pick (Hard)	39%	58% (↑ 19)	52%	55% (↑ 3)
Dual Shoes Place	6%	9% (↑ 3)	22%	24% (↑ 2)
Empty Cup Place	22%	61% (↑ 39)	32%	36% (↑ 4)
Mug Hanging (Easy)	6%	6%	11%	27% (↑ 16)
Mug Hanging (Hard)	1%	1%	4%	4%
Pick Apple Messy	35%	39% (↑ 4)	18%	11% (↓ 7)
Put Apple Cabinet	20%	45% (↑ 25)	34%	55% (↑ 21)
Shoe Place	44%	43% (↓ 1)	52%	58% (↑ 6)
Tool Adjust	54%	42% (↓ 12)	70%	69% (↓ 1)
Average	31.8%	41.6% (↑ 9.8)	36.1%	44.8% (↑ 8.7)

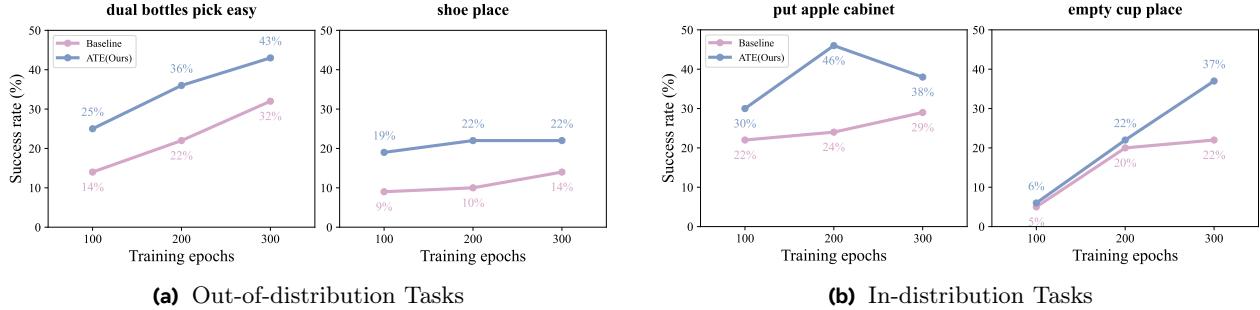


Figure 3 Evaluation of DP baseline with and without ATE across (a) Out-of-distribution tasks and (b) In-distribution tasks. Incorporating ATE consistently improves success rates and accelerates convergence, with the most significant gains observed on challenging tasks where baseline performance is low.

tasks demonstrate consistent improvements with the ATE adaptation method, indicating strong generalization even in highly dynamic simulated environments.

Real-World Results. To further address **Q1** in terms of efficient cross-task and cross-embodiment adaptation, we conduct real-world experiments on a dual-arm RealMan robot, with each arm having 7 DoF and equipped with Agibot OminiPicker single-DoF grippers. To be specific, we design four representative manipulation tasks for our experiments. For each of the four tasks, we collect 160 demonstration trajectories, with the detailed experimental setups and data collection procedures provided in Appendix C and Appendix H. The training process adopts a batch size of 48 for 120k steps, and success rates are measured at 60k, 90k, and 120k steps to assess both convergence speed and final performance. The experiments focus on long-horizon manipulation sequences, each lasting over one minute and requiring precise coordination between arms and objects, thus providing a strong testbed to validate ATE for efficient cross-task

Table 3 Task success rates on ManiSkill3 benchmark.

Task	RDT-1B	RDT-1B + ATE(Ours)
Push Cube	65.2%	78.4% (↑ 13.2)
Pick Cube	7.6%	14.8% (↑ 7.2)
Average	36.4%	46.6% (↑ 10.2)

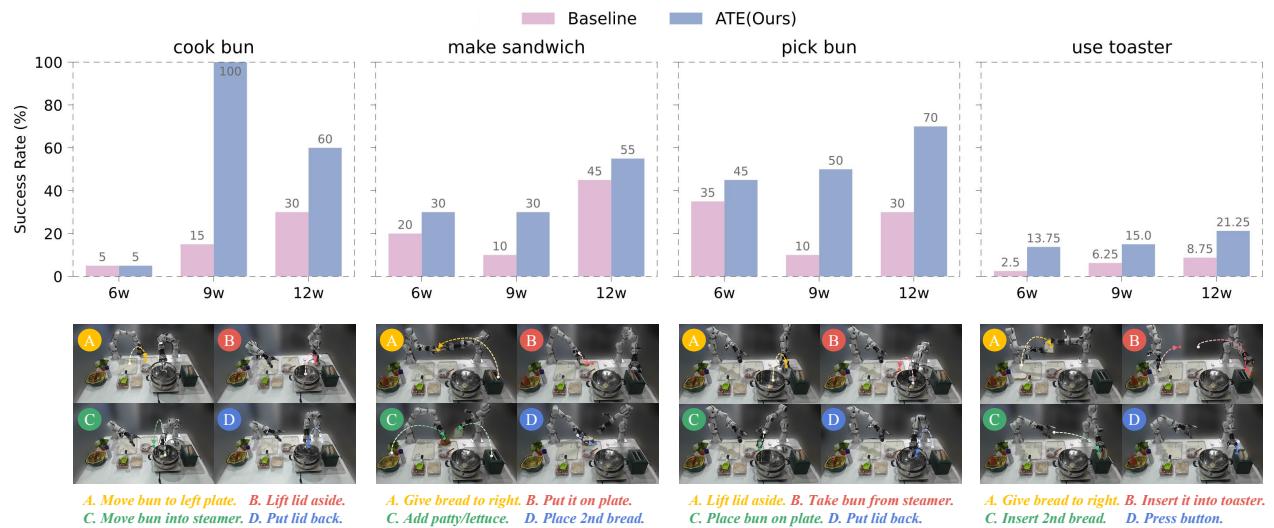


Figure 4 Real-world evaluation on physical robot experiments. Top panel reports success rates across four tasks and overall average, comparing ATE with the baseline π_0 . Bottom panel shows full execution trajectories of four representative tasks, covering long-horizon, single-arm, and dual-arm scenarios.

and cross-embodiment adaptation. The definitions of the four tasks are as follows.

- **Cook Bun:** The right gripper grasps a bun from the right plate and places it on the left plate, while the left gripper lifts the lid and places it aside. The left gripper then picks a bun from the left plate and places it into the steamer, and finally returns the lid to the steamer.
- **Pick Bun:** The left gripper lifts the lid temporarily, grasps the bun farthest from the camera inside the steamer, and places it on the left plate. The lid is then returned to the steamer.
- **Make Sandwich:** The left gripper retrieves one toasted bread slice from the toaster and transfers it to the right gripper, which places it on the left plate. The right gripper then adds the patty, lettuce, and top bread slice handed from the left gripper.
- **Use Toaster:** The left gripper picks up the farther bread slice and passes it to the right gripper for toaster insertion. The remaining slice is inserted similarly, followed by pressing the toaster button.

These tasks involve minute-scale operations that require bimanual collaboration, posing challenges especially with limited adaptation data. The evaluation is run on a single NVIDIA 4090 at 20 Hz, and each task is evaluated over 20 trials, where ‘success’ is defined as completing the long-horizon sequence, including coordinated object manipulation and placement. Initial object positions are randomly sampled within a radius of 3.5 cm. For the *Use Toaster* task, we adopt a graded success metric: fully inserting a bread slice counts as 1.0, tilted insertion as 0.25, and partial insertion as 0.5.

Figure 4 visualizes success rates across tasks at different training steps. Our method significantly accelerates learning, achieving higher success rates with fewer training steps. For example, in *Cook Bun*, the success rate increases to 100% in 90k steps, while the baseline only reaches 15%. In the single-arm *Pick Bun* task, ATE reaches 50% success at 90k steps and 70% at 120k. For dual-arm tasks (*Make Sandwich* and *Use Toaster*), ATE consistently improves performance across all checkpoints. On average across all tasks, the baseline achieves 16.7% overall success, while ATE reaches 58.1% at 120k steps, demonstrating substantial improvement in both final performance and convergence speed. The largest gains are observed in long-horizon tasks like *Pick Bun*, where proper sequencing and coordination are crucial.

In real-world evaluation, we also observe manifest improvements in motion quality. Due to action-space alignment, ATE can generate smoother trajectories compared to the baseline, which often shows abrupt movements and collisions. This is particularly evident in *Cook Bun* and *Pick Bun*, where the baseline applies excessive force to the steamer lid. In contrast, ATE modulates force appropriately, allowing the robot to

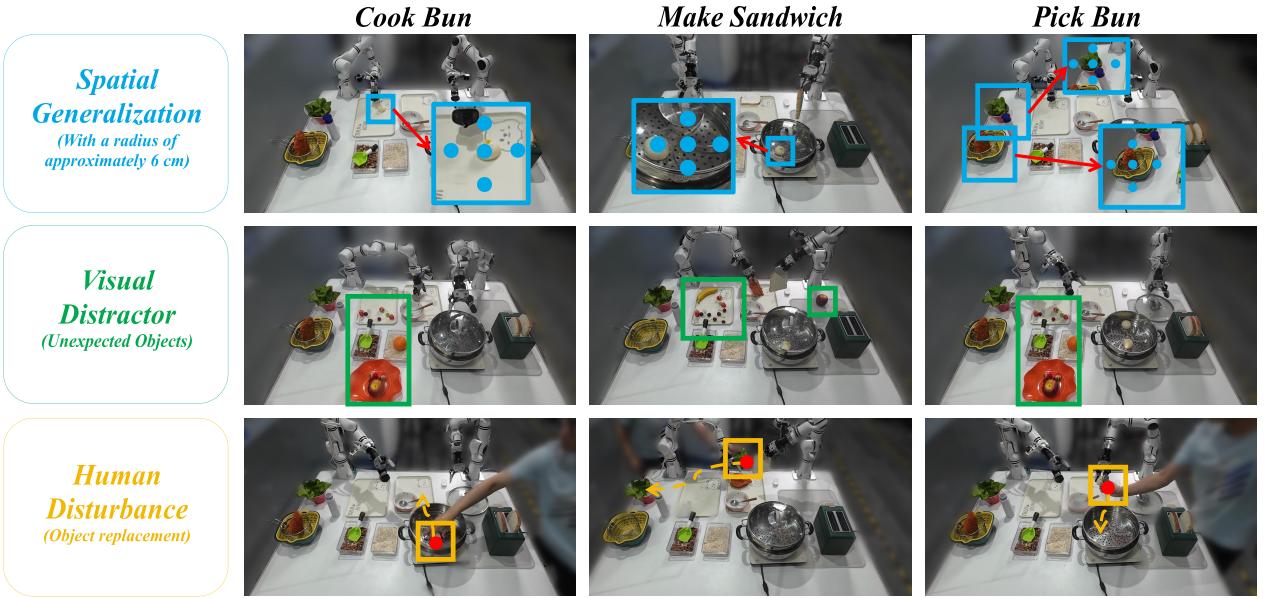


Figure 5 Real-world generalization settings. For the Spatial Generalization experiment, we vary the initial positions of manipulated objects. For the Visual Distractors experiment, we randomly place unseen items (e.g., fruits) in the scene. For the human disturbance experiment, we intervene at key execution points by resetting already completed sub-tasks.

pause and reposition as needed. Here, we use the *Cook Bun* task as a representative example, with all related 6-dimensional force data and analyses provided in Appendix G.

Discussion. In general, simulation and real-world experiments consistently demonstrate that ATE facilitates efficient cross-task and cross-embodiment adaptation. Across RoboTwin 1.0 and ManiSkill3 benchmarks, ATE improves final success rates, accelerates convergence, and enhances sample efficiency compared to baseline policies. Real-world evaluations on the dual-arm RealMan robot further confirm these advantages, with ATE achieving higher success rates, smoother trajectories, and better long-horizon coordination. These results indicate that leveraging a unified action latent space together with latent-space guidance allows the policy to quickly adapt to new tasks and embodiments while preserving pre-trained visuomotor knowledge, effectively addressing Q1.

5.3 Generalization Ability Analysis

To answer Q2, we evaluate the robustness and generalization of ATE under diverse real-world perturbations, including variations in illumination, visual distractions, object morphology, clutter, and human disturbances. Each experiment is evaluated over 5 trials, with success defined as completing the full manipulation sequence. The experimental setups for all perturbations, except for the illumination variation, are illustrated in Fig. 5.

Varying Illumination. We alter the lighting conditions while keeping the color temperature fixed at 5800 K. The illumination levels are set to low, high, and flicker around the optimal value. As shown in Table 4, ATE consistently outperform the baseline, particularly under extreme or flickering illumination, where the baseline often failed. This indicates that ATE is robust to lighting intensity variations, maintaining task performance even under conditions not seen during training.

Spatial Generalization. To further examine position generalization, we extend the randomized initial placement experiments beyond the original 20 trials. The objects are displaced within a larger radius of approximately 6.5 cm, with five trials conducted at each boundary position (up, down, left, right) as well as at the initial location. For the *Pick Bun* and *Cook Bun* tasks, the buns were moved accordingly, while for the *Make Sandwich* task, the meat slices and lettuce were repositioned. Table 5 shows that ATE achieved success rates of 40%, 60%, and 40% on *Cook Bun*, *Pick Bun*, and *Make Sandwich*, respectively. These results demonstrate

ATE’s ability to adapt to larger spatial shifts, highlighting efficient position generalization across diverse manipulation tasks.

Visual Distractor. We introduce irrelevant objects that are not present in the dataset, such as scattered fruits, puzzle pieces, and trays, to place on the table as visual distractions. Table 5 shows that ATE outperformed the baseline in all tasks, with success rates of 40% for *Cook Bun*, 80% for *Pick Bun*, and 40% for *Make Sandwich*. This suggests that ATE effectively focuses on task-relevant objects while ignoring irrelevant clutter, demonstrating strong visual generalization.

Human Disturbance. During the task, when the robotic arm successfully grasped a bun or other objects, a human quickly interfered—for example, removing a bun from the gripper or repositioning ingredients—forcing the policy to reattempt. Table 5 shows that ATE retained 0% success on *Cook Bun*, 40% on *Pick Bun*, and 60% on *Make Sandwich*, whereas the baseline largely failed. These results highlight ATE’s robustness to sudden external interventions, suggesting that the policy can recover from unexpected disturbances and adapt its behavior accordingly.

Table 4 Success rates (%) under different illumination settings (5 trials each).

Method	Cook Bun	Pick Bun	Make Sandwich
Low Illumination Condition			
ATE	60%	40%	40%
Baseline	0%	0%	40%
High Illumination Condition			
ATE	60%	40%	60%
Baseline	0%	0%	40%
Flickering Illumination Condition			
ATE	20%	0%	20%
Baseline	0%	0%	20%

Table 5 Success rates (%) under different generalization challenges (5 trials each).

Challenge	Cook Bun		Pick Bun		Make Sandwich	
	ATE	Baseline	ATE	Baseline	ATE	Baseline
Visual Distractor	40%	20%	80%	20%	40%	40%
Spatial Generalization	40%	0%	60%	40%	40%	20%
Human Disturbance	0%	0%	40%	0%	60%	40%

Discussion Overall, across all generalization experiments—including varying illumination, visual distractors, spatial shifts, human disturbances, and object morphology/clutter variations—ATE consistently outperforms the baseline. These results directly address **Q2**, showing that ATE enhances the robustness of the VLA against diverse real-world perturbations. The strong generalization arises because our method constrains the policy within the structured latent space learned during pre-training, which effectively preserves pre-trained visuomotor priors while allowing adaptation to new environmental variations. By maintaining behavior within this latent manifold, ATE enables the policy to recover from unexpected disturbances, focus on task-relevant features, and adapt to unseen object configurations, demonstrating reliable performance under conditions not encountered during training.

5.4 Ablation Study

To answer **Q3**, we study the impact of the aligned and structured action latent space on cross-embodiment and cross-task adaptation. We compare our two-stage Info-VAE, which embeds adaptation actions into a mode of the pre-trained latent manifold, against a single-stage VAE trained directly on task-specific data, using both π_0 and Diffusion Policy backbones. As shown in Figure 6, the two-stage approach consistently outperforms the single-stage baseline, achieving higher final success rates, especially on long-horizon or challenging tasks. By

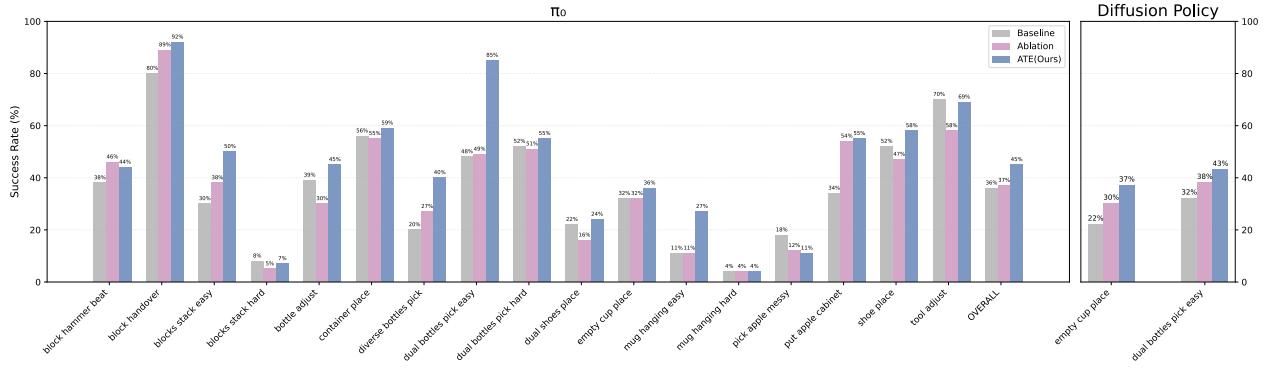


Figure 6 Ablation study on the effect of two-stage Info-VAE training. We compare the proposed two-stage Info-VAE with a single-stage VAE trained only on task-specific datasets, using both π_0 and Diffusion Policy backbones. For the Diffusion Policy results, we evaluate both in-distribution (\dagger) and out-of-distribution generalization settings (\ddagger).

constraining adaptation actions within the structured latent space, our method effectively reduces the domain gap between pre-training and target embodiments, while preserving valuable pre-trained visuomotor priors. These results confirm that the aligned latent space is critical for efficient adaptation, enabling downstream policies to rapidly leverage prior knowledge and generalize across different tasks and robotic platforms.

6 Conclusion

In this work, we present **ATE**, a lightweight and data-efficient framework for adapting VLAs to new embodiments and tasks. By unifying disparate action spaces and steering the pre-trained model via classifier guidance, our method effectively bridges the action distribution gap between pre-training and downstream adaptation. The experimental results on representative VLA architectures, including DP, RDT, and π_0 verify the widely applicability of our adaptation method. The cross-task adaptation experiments on RoboTwin and ManiSkill benchmarks show that **ATE** consistently outperforms baselines. By constructing complex bimanual collaboration tasks in a new robot platform, we observe **ATE** obtains significantly better performance than standard adaptation methods, especially for spatial, visual, and disturbance generalization settings.

Our results highlight the practicality and scalability of **ATE**, offering a plug-and-play solution that seamlessly integrates with diffusion- and flow-based VLAs without altering their architectures. Looking ahead, we envision that this alignment-and-guidance paradigm can serve as a foundation for fine-tuning generalist robot policies at scale, enabling efficient adaptation to new robotic platforms and diverse tasks.

References

- Johannes Anschütz and Arthur-César Le Bras. Prismatic dieudonné theory. In *Forum of Mathematics, Pi*, volume 11, page e2. Cambridge University Press, 2023.
- Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.
- Erik Bauer, Elvis Nava, and Robert K Katzschmann. Latent action diffusion for cross-embodiment manipulation. *arXiv preprint arXiv:2506.14608*, 2025.
- 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.
- Hongzhe Bi, Lingxuan Wu, Tianwei Lin, Hengkai Tan, Zhizhong Su, Hang Su, and Jun Zhu. H-rdt: Human manipulation enhanced bimanual robotic manipulation, 2025. URL <https://embodiedfoundation.github.io/hrdt>.
- Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- Johan Bjorck, Fernando Castañeda, Nikita Cherniakov, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr0ot n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- 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.
- Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2502.14420*, 2025.
- Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.
- Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023a.
- Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18000–18010, 2023b.
- Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham,

Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. <https://arxiv.org/abs/2310.08864>, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 258–267, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. *arXiv preprint arXiv:2503.19757*, 2025.

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.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL <https://openreview.net/forum?id=8kbp23tSGYv>.

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.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024. URL <https://openreview.net/forum?id=ZMnD6QZAE6>.

- Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.
- Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015.
- Zhixuan Liang, Yao Mu, Yixiao Wang, Tianxing Chen, Wenqi Shao, Wei Zhan, Masayoshi Tomizuka, Ping Luo, and Mingyu Ding. Dexhanddiff: Interaction-aware diffusion planning for adaptive dexterous manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1745–1755, 2025.
- Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=PqvMRDCJT9t>.
- Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.
- Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.
- Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023a. URL <https://openreview.net/forum?id=XVjTT1nw5z>.
- Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=XVjTT1nw5z>.
- Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.
- Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27649–27660, 2025.
- NVIDIA. Gr00t n1: An open foundation model for generalist humanoid robots, 2025.
- Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.
- Mathis Petrovich, Michael J Black, and Gülcin Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10985–10995, 2021.
- Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.
- Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=PxTIG12RRHS>.
- 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.
- Octo Model Team, Dibya Ghosh, Homer Walké, 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.

- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025a.
- Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025b.
- Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yixin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *Forty-second International Conference on Machine Learning*, 2025c.
- Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv preprint arXiv:2502.02175*, 2025.
- Seonghyeon Ye, Joel Jang, Byeonguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.
- Haoqi Yuan, Bohan Zhou, Yuhui Fu, and Zongqing Lu. Cross-embodiment dexterous grasping with reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=twIPsX9qHn>.
- Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *arXiv preprint arXiv:2503.20384*, 2025a.
- Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *arXiv preprint arXiv:2503.20384*, 2025b.
- Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.
- Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22508–22519, 2025.
- Yifan Zhong, Xuchuan Huang, Ruochong Li, Ceyao Zhang, Yitao Liang, Yaodong Yang, and Yuanpei Chen. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. *arXiv preprint arXiv:2502.20900*, 2025.
- 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.

A Additional Related Works

Constructing Unified Action Space. Establishing a shared discrete token dictionary to discretize continuous robotic actions is a common approach for mapping heterogeneous actions into a shared space (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024; Pertsch et al., 2025). Furthermore, UniVLA (Bu et al., 2025) leveraged a Vector-Quantized VAE to extract task-centric latent actions from massive cross-embodiment videos for VLA pre-training and post-training. Similarly, UniACT (Zheng et al., 2025) also learned a discrete atomic behavior codebook from large-scale robotic data to form a universal discrete latent action representation. However, these methods primarily focus on establishing a universal latent representation for actions across different embodiments, failing to resolve the significant distribution discrepancy that persists between pre-training and adaptation data within the latent space. In contrast, we approach this issue from the perspective of bridging the distribution gap. Our method constructs a unified action space by first encoding the pre-training actions to form a base latent distribution, and subsequently embedding the adaptation actions into the modes of this pre-training action distribution, which results in a structured latent space. While Bauer et al. (2025) combined separate VAEs with contrastive learning to map different embodiment-specific instances of the same action into a shared latent space, their method’s heavy reliance on a hand motion retargeting system to generate these instances makes it challenging to apply in common VLA pre-training and post-training recipe.

B Additional Real-World Experiments: Tool Use

B.1 Experimental Setup

To validate our method on the tool-use VLA challenge, we collected an additional 80 trajectories and designed the **Make Yogurt Bowl** task. This task requires using both a *shovel* and a *spoon* for precise ingredient manipulation.

Data. 80 high-quality trajectories are collected for this task, covering variations in initial object placement.

Platform. All experiments are conducted on a dual-arm **RealMan** robot, where each arm has 7 DoF and is equipped with Agibot OminiPicker single-DoF grippers. The left arm handles the spatula, and the right arm handles the spoon.

Training. Models are trained for 120k steps on 8 NVIDIA A100 GPUs with a batch size of 12 per GPU. Execution runs at 20 Hz on a single NVIDIA 4090.

Task Description. Use the right gripper to pick up the shovel, scoop out the nuts and oats, and then pour yogurt into the bowl. During the process, use the left gripper to stir with a spoon.

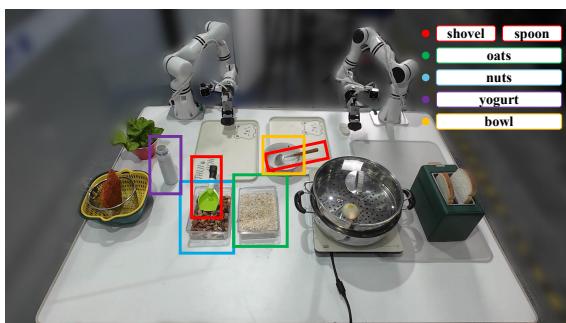
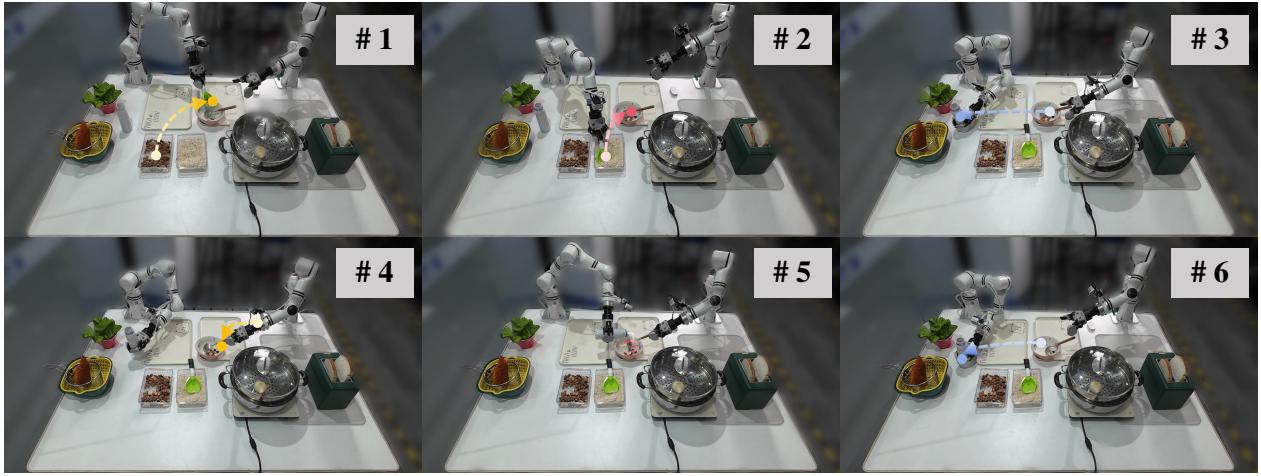


Figure 7 Experimental setup for the *Make Yogurt Bowl* task. The dual-arm robot manipulates ingredients using a spatula (right hand) and a spoon (left hand), demonstrating precise coordination. Key objects are highlighted.

Table 6 Success rate (%).

Task	Baseline	ATE(Ours)
Make Yogurt Bowl	15%	25%



#1: Pick up the shovel with the right gripper. Scoop the nuts using the shovel. #2: Scoop the oats using the shovel. #3: Pick up the yogurt bottle with the right gripper. #4: Pick up the spoon with the left gripper. #5: Pour the yogurt into the bowl while stirring with the spoon in the left gripper. #6: Place the yogurt bottle back in its original position.

Figure 8 Real-world execution of the Make Yogurt Bowl task.

B.2 Results.

For the *Make Yogurt Bowl* task, the robot must perform highly coordinated dual-arm manipulation: the right gripper picks up a spatula to scoop nuts and oats and pour yogurt into the bowl, while the left gripper continuously stirs the mixture with a spoon. This task requires precise control of multiple tools simultaneously, careful timing to avoid spillage, and maintaining proper spatial relationships between ingredients and utensils. Such multi-tool coordination represents a challenging test for vision-language-action policies, as small deviations can lead to task failure.

As shown in Table 6, our ATE method achieves a success rate of 25%, outperforming the baseline at 15%, with each method tested over 20 trials. This improvement highlights ATE’s ability to leverage pre-trained latent representations and structured visuomotor priors, enabling more reliable tool handling and fine-grained manipulation. The results indicate that ATE effectively captures the dependencies between dual-arm actions and tool interactions, allowing it to generalize across subtle variations in object positions and motions, whereas the baseline struggles with the complexity of simultaneous tool use.

B.3 Visual Distractor Generalization

To evaluate visual distractor robustness, unexpected objects (e.g., extra fruit pieces, small puzzles) are randomly placed on the table. The robot must complete the task while ignoring these unexpected items. Success is defined as correctly completing all phases with proper tool use. Each experiment is repeated 5 times, and the results are summarized in Table 7.

C Experimental Setup

C.1 Fast Adaptation in RoboTwin/Maniskill

Each model is trained and evaluated across a range of manipulation tasks using the specified number of demonstrations. All experiments are conducted using multi-GPU training setups.

RoboTwin 1.0 Benchmark RoboTwin 1.0 is adopted as the core evaluation benchmark. As a generative digital twin framework tailored for dual-arm tasks, RoboTwin 1.0 integrates 3D generative models and large

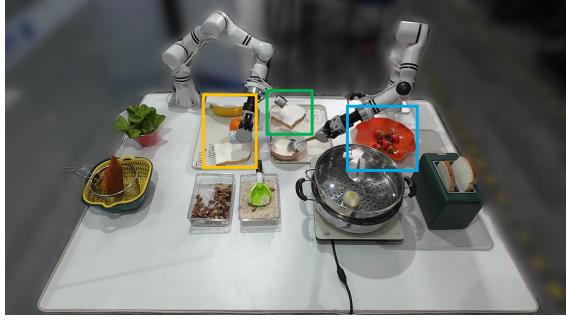


Figure 9 Experimental setup of the *Make Yogurt Bowl* task under visual distractors. Additional irrelevant objects (e.g., fruit pieces) are placed on the table, while the dual-arm robot must focus on manipulating the spoon and spatula to complete the task.

language models to synthesize diverse object-centric scenes along with spatially grounded robot code. A total of 17 tasks are selected, spanning both single-arm and dual-arm manipulation scenarios (e.g., tool adjustment, dual-bottle picking).

ManiSkill3 Benchmark In addition, we use ManiSkill3, a fast, GPU-parallelized simulator designed for contact-rich single-arm manipulation. We evaluate our method on 2 representative tasks such as push cube and pick cube, which emphasize physical contact and fine-grained control.

Simulation Training Configuration. We summarize the training settings for each policy in Table 8.

Table 8 Training configurations for different policy models and environments. BS/GPU denotes batch size per GPU.

Policy	Benchmark	BS/GPU	GPUs	Training Duration	Dataset Size
RDT	RoboTwin 1.0 (17 tasks, , multi-task)	16	4 A100	100k steps	100/task
	ManiSkill3 (2 tasks, multi-task)	12	2 A100	100k steps	100/task
π_0 (PyTorch)	RoboTwin 1.0 (17 tasks, multi-task)	12	4 A100	60k steps	50/task
	RoboTwin 1.0 (5-task pretrain)	128	1 A100	2000 epochs	50/task
DP	RoboTwin 1.0 (Finetune)	128	1 A100	300 epochs	50/task

Diffusion Policy Evaluation. To assess the effectiveness of our proposed ATE training strategy, we perform evaluations under both in-distribution and out-of-distribution settings using Diffusion Policy as the backbone model:

- **Pretraining:** A base model is trained on a fixed set of 5 RoboTwin 1.0 tasks.
- **Evaluation:**
 - *In-distribution setting:* We fine-tune and evaluate the model on a subset of the same tasks used in pretraining to assess in-distribution task adaptation.
 - *Out-of-distribution setting:* We fine-tune and evaluate the model on novel RoboTwin 1.0 tasks that were not part of the pretraining set, measuring generalization to unseen tasks.

This setup allows us to verify whether our ATE policy fine-tuning improves generalization and adaptation performance across both familiar and novel tasks.

Table 7 Success rate (%) under Visual Distractors.

Task	Baseline	ATE(Ours)
Make Yogurt Bowl	0%	20%

C.2 Fast Adaptation to Dual Realman Arms in Real World

Data. We collect 160 high-quality trajectories for each of the four representative long-horizon manipulation tasks (*Cook Bun*, *Pick Bun*, *Make Sandwich*, *Use Toaster*), each lasting over one minute and requiring precise bimanual coordination between arms and objects.

Platform and Vision Setup. All experiments are conducted on a dual-arm **RealMan** robot, where each arm has 7 DoF and is equipped with Agibot OminiPicker single-DoF grippers. The visual system consists of a main overhead camera (Intel RealSense L515, RGB only, 960×540, 30 Hz, automatic exposure) and two wrist-mounted cameras (Intel RealSense D405, RGB only, 960×540, 30 Hz, automatic exposure) to capture the workspace from multiple perspectives.

Training. Models are trained for 120k steps using 8 NVIDIA A100 GPUs, with a batch size of 12 per GPU.

Task Setup. The key objects involved in the four representative long-horizon manipulation tasks are illustrated in Figure 10.

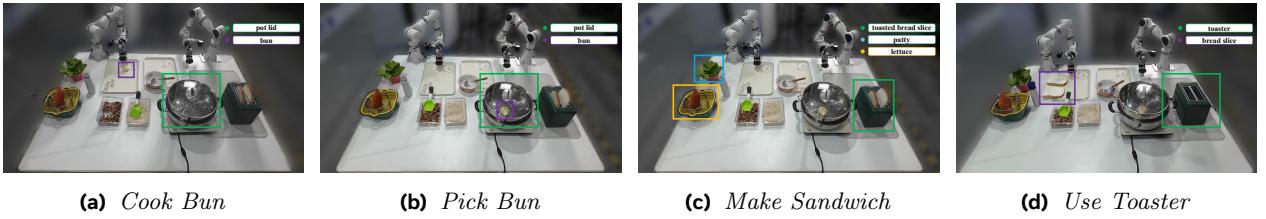


Figure 10 Key objects involved in the four representative long-horizon manipulation tasks.

C.3 Generalization Ability Evaluation on Dual Realman Arms

This section provides essential details to reproduce the generalization experiments in Section 5.3, focusing on illumination settings.

Illumination Settings We used a set of controllable studio lamps to vary lighting conditions. All lamps have a fixed color temperature of 5800 K. All illumination settings are illustrated in Figure 11.

- **Lamp:** COB studio fill light
- **Low Illumination:** 25 lux
- **High Illumination:** 65 lux
- **Flickering Illumination:** alternating between 0 and 45 lux, flicker frequency 2 Hz

Notes: All experiments were conducted under the same camera configuration to ensure consistent observation inputs for the policy.

D Training Hyperparameters

We summarize the key hyperparameter settings used for both stages of our method, including (i) Learning the Unified Action Latent Space, and (ii) Steering Efficient Adaptation with Latent Guidance.

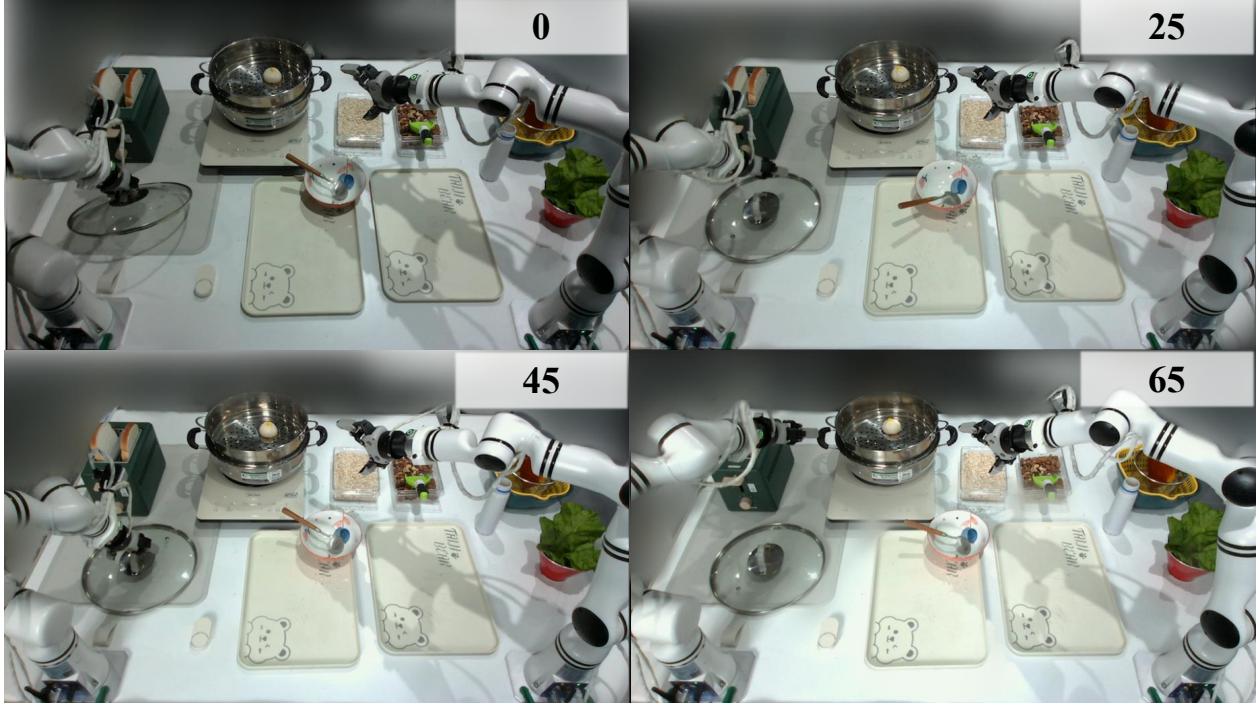


Figure 11 Illustration of the illumination setup under three conditions: low (25 lux), high (65 lux), and flickering (alternating between 0 and 45 lux at 2 Hz).

Table 9 Hyperparameters for Info-VAE pretraining.

Hyperparameter	Value
Latent dimension	512
Temporal input length	64 (RDT), 50 (π_0), 14 (DP)
Input dimensionality	128 (RDT), 32 (π_0), 14 (DP)
Optimizer	Adam
Learning rate	1×10^{-4}
Weight decay	1×10^{-4}
Batch size	64

Table 10 Task-specific fine-tuning (RoboTwin).

Hyperparameter	RDT	π_0	DP
Chunk size	64	50	8
Batch size	64	24	128
Learning rate	1.0e-4	2.5e-5	1.0e-4
Training duration	100k steps	60k steps	300 epochs

Other platforms. For ManiSkill and Real Robot experiments, we use the following task-specific fine-tuning settings:

- **ManiSkill (simulator):** Chunk size 64 (RDT), 24 batch size, learning rate 1.0e-4, training duration 100k steps.
- **Real Robot (physical):** Chunk size 20 (π_0), batch size 48, learning rate 2.5e-5, training duration 120k steps.

E Algorithmic Description of ATE

Algorithm 1 Stage 1: Learning the Unified Action Latent Space

Input: Pre-training dataset $\mathcal{D}_{\text{pretrain}}$, Adaptation dataset $\mathcal{D}_{\text{adaptation}}$, Pre-training action VAE ϕ , Adaptation action VAE ψ , Number of epochs N

// Learn the pre-training action VAE

for epoch = 1 to N **do**

- Sample a batch of actions $\{\mathbf{a}_{t:t+H-1}^{(i)}\}_{i=1}^n$ from $\mathcal{D}_{\text{pretrain}}$
- Update pre-training action VAE ϕ via maximizing $\mathcal{L}(\phi; \mathcal{D}_{\text{pretrain}})$ in Eq. (5)

end for

// Calculate the pre-training action latent distribution

Initialize the running mean μ_ϕ and variance Σ_ϕ of the pre-trained action latent distribution $q_\phi(z)$

for all possible sample $\mathbf{a}_{t:t+H-1}$ in **do**

- Encode the latent z of the action chunk $\mathbf{a}_{t:t+H-1}$ via ϕ
- Update the running mean μ_ϕ and variance Σ_ϕ

end for

// Learn the adaptation action VAE

for epoch = 1 to N **do**

- Sample a batch of actions $\{\mathbf{a}_{t:t+L-1}^{(i)}\}_{i=1}^n$ from $\mathcal{D}_{\text{adaptation}}$
- Update pre-training action VAE ψ via maximizing $\mathcal{L}(\psi; \mathcal{D}_{\text{adaptation}}, \phi)$ in Eq. (6)

end for

Algorithm 2 Stage 2: Steering Adaptation of **Flow-based VLAs** with Latent Guidance

Input: Flow-based VLA v_θ , Adaptation dataset $\mathcal{D}_{\text{adaptation}}$, Adaptation action encoder \mathbf{E}_ψ , Guidance scale λ , Learning rate η

repeat

- Sample an observation-language-action pair $(\mathbf{o}_t, l, \mathbf{a}_{t:t+L-1}) \sim p_{\mathcal{D}_{\text{adaptation}}}(\cdot)$
- Sample $\epsilon \sim \mathcal{N}(0, I)$, $\tau \sim \mathcal{U}([0, 1])$
- Corrupt clean action chunk $\mathbf{a}_{t:t+L-1}^\tau = \tau \mathbf{a}_{t:t+L-1} + (1 - \tau)\epsilon$
- // Compute the guidance
- Get the latents $\hat{z} = \mathbf{E}_\psi(\mathbf{a}_{t:t+L-1}^\tau)$, $z = \mathbf{E}_\psi(\mathbf{a}_{t:t+L-1})$ of noisy action chunk and clean action chunk
- Obtain the latent guidance $g = -\nabla_{\mathbf{a}_{t:t+L-1}^\tau} \|z - \hat{z}\|^2$
- // Steer the fine-tuning
- Get the ground truth velocity $v_t = \mathbf{a}_{t:t+L-1} - \epsilon$
- Update the VLA via $\theta = \theta - \eta \nabla_\theta \| [v_\theta(\mathbf{a}_{t:t+L-1}^\tau; \tau, \mathbf{o}_t, l) + \frac{1-\tau}{\tau} \lambda g] - v_t \|^2$

until Converged

Algorithm 3 Stage 2: Steering Adaptation of **Diffusion-based VLAs** with Latent Guidance

Input: Diffusion-based VLA ϵ_θ , Adaptation dataset $\mathcal{D}_{\text{adaptation}}$, Adaptation action encoder \mathbf{E}_ψ , Guidance scale λ , Learning rate η

repeat

- Sample an observation-language-action pair $(\mathbf{o}_t, l, \mathbf{a}_{t:t+L-1}) \sim p_{\mathcal{D}_{\text{adaptation}}}(\cdot)$
- Sample $\epsilon \sim \mathcal{N}(0, I)$, $k \sim \mathcal{U}(\{1, 2, \dots, T\})$
- Corrupt clean action chunk $\mathbf{a}_{t:t+L-1}^k = \sqrt{\bar{\alpha}_k} \mathbf{a}_{t:t+L-1} + \sqrt{1 - \bar{\alpha}_k} \epsilon$
- // Compute the guidance
- Get the latents $\hat{z} = \mathbf{E}_\psi(\mathbf{a}_{t:t+L-1}^k)$, $z = \mathbf{E}_\psi(\mathbf{a}_{t:t+L-1})$ of noisy action chunk and clean action chunk
- Obtain the latent guidance $g = -\nabla_{\mathbf{a}_{t:t+L-1}^k} \|z - \hat{z}\|^2$
- // Steer the fine-tuning
- Get the ground truth velocity $v_t = \mathbf{a}_{t:t+L-1} - \epsilon$
- Update the VLA via $\theta = \theta - \eta \nabla_\theta \|[\epsilon_\theta(\mathbf{a}_{t:t+L-1}^k; k, \mathbf{o}_t, l) - \sqrt{1 - \bar{\alpha}_k} \lambda g] - \epsilon\|^2$

until Converged

F Hardware Details

We present the dual-arm platform constructed and utilized in this work. Two identical Realman RM75-6s robotic arms are mounted on a white lifting desk using four G-type fixtures. At their end-effectors, we fixed the OminiPicker grippers developed by AGIBOT together with RealSense-D405 short-range depth cameras. Both grippers communicate with the workstation through USB-to-serial protocols. In addition to the two cameras mounted on the robotic arm end-effectors, we also strategically positioned one RealSense-L515 high-precision depth camera on a tripod, positioned appropriately behind the arms to capture a complementary global view. All three cameras are directly connected to our workstation via USB 3.2 interfaces. The two robotic arms are connected to the same local area network as the workstation and communicate using the Ethernet-based SDK provided by Realman. An overview of the hardware setup is illustrated in Figure 12, while the corresponding hardware specifications are listed in Table 11 for reference.

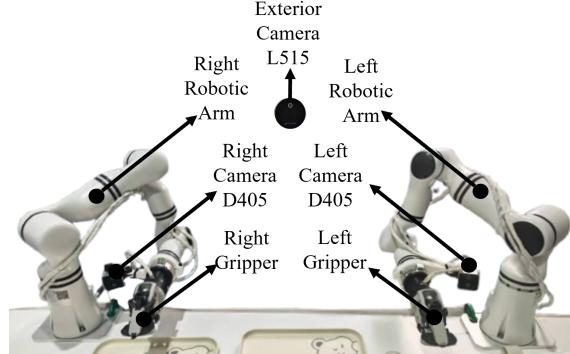
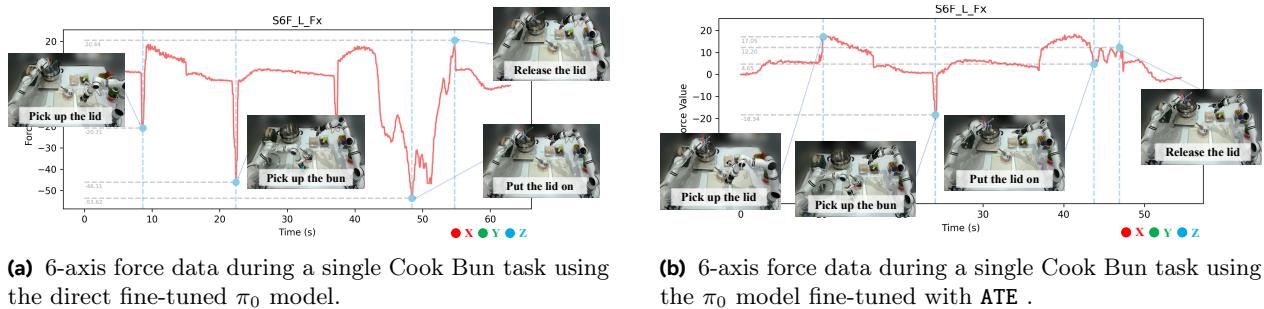


Figure 12 Hardware feature. The wrist-mounted camera is the RealSense D405 model, with an optimal operating range of 7cm – 50cm, while the exterior camera is the RealSense L515 model, with an optimal operating range of 0.25m – 9m. The two robotic arms are of exactly the same model rather than mirror versions.

Parameter	Value (%)
DoF	$8 \times 2 = 16$
Arm weight	$7.9 \times 2 = 15.8$ kg
Arm Payload	5000g
Arm reach	638.5mm
Arm Repeatability	0.05mm
Joint motion range	J1 $\pm 178^\circ$, J2 $\pm 130^\circ$, J3 $\pm 178^\circ$, J4 $\pm 135^\circ$, J5 $\pm 178^\circ$, J6 $\pm 128^\circ$, J7 $\pm 360^\circ$
Gripper range	120mm
Gripper weight	0.43kg
Gripper Payload	1.5kg
Gripper max force	30N

Table 11 Technical specifications. In this table, the term arm refers to the Realman RM75-6s model, while gripper refers to the Ominipicker developed by AGIBOT.

G The Hidden Advantage of ATE: Safer and Smoother Manipulation



(a) 6-axis force data during a single Cook Bun task using the direct fine-tuned π_0 model.

(b) 6-axis force data during a single Cook Bun task using the π_0 model fine-tuned with ATE .

Figure 13 Six-axis force comparison during the Cook Bun task. (a) Using the direct fine-tuned π_0 model, the force signals exhibit instability and less structured patterns. (b) Using the π_0 model fine-tuned with ATE, the force signals are smoother and more consistent, indicating improved robustness and better adaptation to the target embodiment.

While our primary objective was to facilitate data-efficient adaptation by bridging the action distribution gap, our experiments revealed a **valuable and unanticipated benefit**: superior motion smoothness and more stable force control. We validate this finding through a detailed analysis of six-axis force measurements. The term six-axis force refers to forces and torques expressed in the coordinate frame rigidly attached to the target object: the three force components are along the positive directions of the coordinate axes, while the remaining three correspond to torques around these axes.

When performing the cook bun task with the direct fine-tuned policy π_0 , we observed that the robot arm frequently exerts excessive force while placing the steam lid back on the steamer, which often led to deformation of the steamer. By contrast, when the same action was executed by the fine-tuned policy π_0 with ATE, the deformation of the steamer was significantly mitigated.

To quantitatively substantiate this observation, we used the built-in six-axis force sensor integrated into the robot's end-effector to collect data during task execution. As shown in Figure 13a and Figure 13b, we selected the force component along the x-axis as the primary evaluation criterion. This is because we confirmed that, when placing the lid, the force transmitted from the lid to the gripper is aligned with the x-axis direction. The data presented in the figure are relative measures, obtained by subtracting the sensor readings at the initial pose from the raw sensor values.

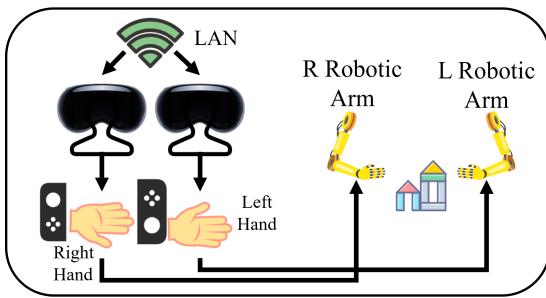
From the figures, it can be observed that the force curve of ATE exhibits fewer sharp fluctuations compared to that of the baseline and remains at a smaller overall magnitude. More specifically, across four representative

peaks and valleys, the baseline policy demonstrates a tendency to apply excessive and unnecessary downward force during the bun-grasping and lid-placing actions.

This finding suggests that by constraining the fine-tuning process within the pre-training action latent space, our method implicitly guides the policy toward safer and more robust physical interactions.

H Real-world Data Collection Details

In this section, we provide a detailed description of how our dataset is collected. As illustrated in Figure 14b, our real-world tasks involve bimanual simultaneous actions. If we were to rely on traditional single-operator teleoperation, accomplishing such tasks would impose excessive demands on the operator. To address this challenge, we adopt a dual-operator collaborative teleoperation approach for data collection.



(a) Dual-arm collaborative teleoperation.



(b) Bimanual task involves simultaneous actions.

Figure 14 The setup of real-robot data collection. (a) The overview of how we implement dual-arm data Collection. During teleoperation, the operator holds Joycon with the hand which not cared about by our program. (b) An example that requires both hands to perform logically disparate tasks, simultaneously occurs in the *use toaster* task.

In the dual-operator teleoperation setup, each operator is first equipped with a separate Vision Pro headset, both of which are connected to the same local area network. We then employ the open-source project TrackingStreamer to extract hand-keypoint estimations from the Vision Pro devices and stream them to our workstation. Note that in the wrist pose estimated by Vision Pro, the attached coordinate frame on the wrist differs in orientation from the coordinate frame defined on the Realman robotic arm. To ensure consistency, we apply a right-multiplication with a predefined rotation matrix T_{rotation} to the wrist frame, so that when both the wrist and the robotic arm are in their initial poses, the directions of their coordinate frames are aligned.

After both operators place their corresponding hands (which control the robotic arms) in the initial positions and press the initialization button, the program immediately records the wrist estimation from Vision Pro at that moment, and treats it as the initialization matrix for the current data collection session. Next, we assume the existence of a transformation matrix, such that left-multiplying the initialization matrix with $T_{\text{transform}}$ maps it to the robotic arm coordinate system, perfectly aligning it with the initial pose of the end-effector. By right-multiplying both sides of this equation with the inverse of the initialization matrix, we can solve for $T_{\text{transform}}$

$$T_{\text{transform}} = T_{\text{ee_init}}(T_{\text{vision_init}}T_{\text{rotation}})^{-1}$$

where $T_{\text{ee_init}}$ denotes the initial pose of the end effector in the robot coordinate system, and $T_{\text{vision_init}}$ denotes the wrist pose estimated by Vision Pro at the time of initialization. For every subsequent step, Vision Pro's wrist estimation is first multiplied by the right T_{rotation} and then by the left $T_{\text{transform}}$. This way, the motion and orientation of the hand can be directly mapped into the robotic arm coordinate system,

representing the desired end effector pose. By employing this mapping approach for teleoperation, it is only necessary to know the initial poses of the robotic arm and the initial pose of the interface to be used as the teleoperation reference. Moreover, under the requirements of our experiments, this method enables a clear separation of the two different robotic arms, thereby facilitating the realization of dual operator teleoperation. Next, we utilized the Pinocchio library to solve the inverse kinematics to get the joint angles, which are then transmitted directly to the robot for execution. Beyond using Vision Pro as our primary sensing hardware, we integrate Nintendo Joy-Con devices: operators can use the Joy-Cons to control the gripper's open and close actions, thereby completely eliminating potential misrecognition issues since we use vision-based solutions. Additionally, Joy-Cons serve as auxiliary controllers to notify the operator of code errors, quickly set the robot arms, and confirm dataset recording, among other functions.

The advantages of employing dual operator teleoperation can be outlined as follows. First, tasks requiring both arms to perform different actions simultaneously can be executed effortlessly, enabling the policy which fine-tuned on our dataset to demonstrate superhuman multitasking capabilities. Second, when one arm is idle, its wrist-mounted camera can be aimed at the other working arm, thereby providing three distinct visual perspectives for the active robotic arm. Third, operators are granted greater freedom of movement: they can walk around to achieve larger control ranges for their respective arms without worrying about the state of the other arm. Furthermore, since the two operators' hands are spatially separated, the visual occlusion problem in HandOver-type tasks is naturally resolved. Finally, each of the operator only needs to master half of the skill set, which accelerates skill transfer when switching to new tasks and reduces operator workload.