FASTOPD: ON-POLICY DISTILLATION FOR LIGHTWEIGHT VLA DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) foundation models have scaled rapidly to enhance manipulation performance and generalizability, but this scaling incurs high computational costs that render real-world deployment increasingly challenging.

Existing approaches typically mitigate this issue by designing smaller architectures or reducing the iterative denoising steps in flow-based policies. In this work, we propose *FastOPD*, a foundation-to-lightweight VLA framework that enables the practical deployment of large-scale VLAs through efficient on-policy distillation. Specifically, FastOPD adapts a flow map for single-state teacher supervision and combine it with a self-consistency objective to construct a compact student that learns the teacher dynamics. Furthermore, we theoretically demonstrate that minimizing this objective allows the distilled student to recover a distribution on par with that induced by an ideal few-step teacher model. We evaluate FastOPD across diverse foundation policies in simulation and real-world experiments. On LIBERO, FastOPD retains 84% of the performance of $\pi_{0.5}$ with only two inference steps, reducing inference latency by 78.1% while outperforming existing few-step distillation baselines in average success rate. With LingBot-VLA as the teacher, FastOPD improves the single-step success rate over the base student by 15.9 percentage points on RoboTwin 2.0. We further demonstrate its applicability to a World Action Model (WAM) and deploy a compact student distilled from MolmoAct2 on a real robot.

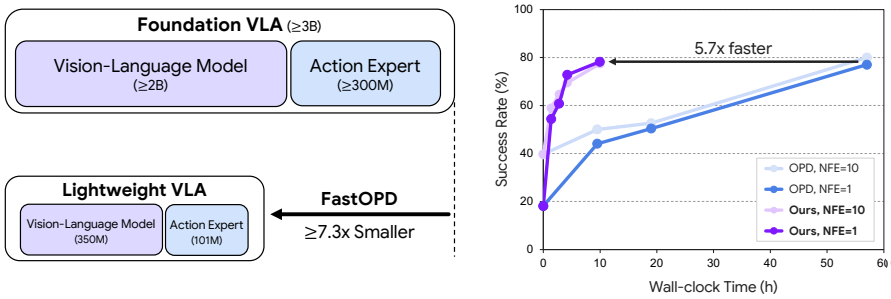
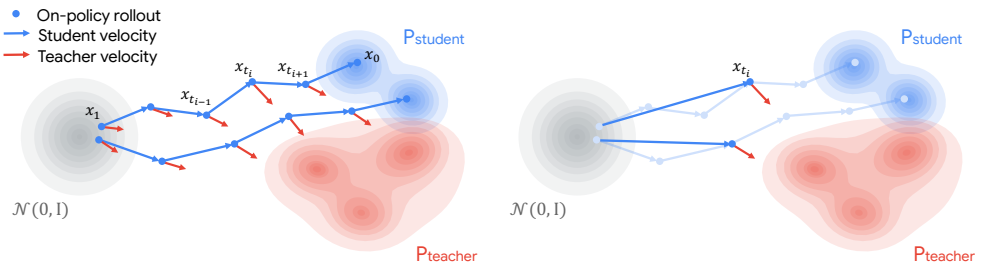


Figure 1: **Foundation-to-lightweight VLA policy distillation with FastOPD.** (Left) Our method enables a small VLA to inherit large-scale VLA knowledge. (Right) FastOPD reaches comparable performance to conventional on-policy distillation (Li et al., 2026a) 5.7× faster.

1 INTRODUCTION

Recent progress in robot learning has moved toward large-scale foundation policies, ranging from Vision-Language-Action (VLA) models (Collaboration, 2024; Black et al., 2025b;a; Bjorck et al., 2025; Wu et al., 2026; Fang et al., 2026a) to emerging World Action Models (WAMs) (Kim et al., 2026; Ye et al., 2026; Yuan et al., 2026). In particular, recent methods such as DreamZero (Ye et al., 2026) and Cosmos Policy (Kim et al., 2026) leverage pretrained video generation models to capture rich visual dynamics and achieve strong generalization and manipulation performance.

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



(a) DiffusionOPD (b) FastOPD (Ours)

Figure 2: **Comparison between DiffusionOPD and FastOPD.** (a) DiffusionOPD evaluates teacher at every denoising step. (b) FastOPD evaluates teacher at a single sampled state reached by a flow-map jump from noise. Blue arrows denote student transitions; red arrows denote teacher velocities.

However, these powerful foundation priors also lead to increasingly large policy backbones and high computational cost, creating a tension between model capability and deployment efficiency on physical robots. Lightweight VLA models such as SmolVLA (Shukor et al., 2025) address the efficiency side of this trade-off by directly designing smaller policies with less than 0.5B parameters.

A complementary direction is to transfer the capability of a powerful foundation policy into a compact student through knowledge distillation. Knowledge Distillation (KD) (Hinton et al., 2015; Mansourian et al., 2025) has been widely studied for transferring the capability of a large teacher model into a smaller student. Specifically, On-Policy Distillation (OPD) (Agarwal et al., 2024) supervises the student using teacher predictions on its own generated trajectories, reducing distribution mismatch and compounding errors from offline distillation. OPD also provides dense supervision along the student trajectory, in contrast to the sparse rewards commonly used in reinforcement learning. While OPD has been actively explored in LLM post-training, recent works have extended it to diffusion and flow models (Li et al., 2026a; Fang et al., 2026b). However, directly applying OPD to flow-based policies is computationally inefficient. It requires rolling out the student's denoising trajectory and repeatedly evaluating the teacher at the visited timesteps, resulting in high training cost for large foundation teachers.

In this work, we revisit few-step flow-map models through the lens of on-policy distillation (OPD) and introduce **FastOPD**, a foundation-to-lightweight VLA framework for efficient policy distillation. Our goal is to transfer the capabilities of a large-scale VLA into a compact student while substantially reducing both training and inference costs, without sacrificing policy performance (see Fig. 1). To reduce the training cost of conventional OPD, FastOPD combines on-policy teacher supervision of the student's velocity field with a self-consistency objective (Frans et al., 2025). Rather than querying teacher supervision throughout an entire generation trajectory, the self-consistency objective propagates local velocity supervision to finite-interval shortcut transitions, as illustrated in Fig. 2. This allows the student to approximate the teacher dynamics without explicitly constructing or evaluating a teacher flow map over each shortcut transition. We further show theoretically that FastOPD implicitly recovers the distribution induced by the teacher's ideal shortcut model, providing a distributional justification for few-step distillation.

Importantly, the same self-consistency formulation naturally induces an any-step flow map (Boffi et al., 2024; Song et al., 2023; Kim et al., 2024; Geng et al., 2025; Sabour et al., 2025), allowing the distilled student to operate with substantially fewer sampling steps at inference. FastOPD therefore compresses the foundation VLA along two complementary dimensions: model capacity and sampling complexity, enabling deployment on resource-constrained, consumer-grade hardware. The proposed formulation also enables stable training in the challenging few-step regime without adversarial objectives or costly higher-order derivative computations. A concurrent work, VLA-OPD (Zhong et al., 2026), also studies on-policy distillation for VLA models, but primarily uses expert supervision to improve policy performance. In contrast, FastOPD focuses on transferring a foundation policy into a lightweight few-step student.

We evaluate FastOPD on two simulation benchmarks and one real-world manipulation task, using diverse foundation teachers with 3B–6B parameters (Black et al., 2025a; Yuan et al., 2026; Fang et al., 2026a). To maximize both architectural and inference efficiency, we adopt SmolVLA as the lightweight student policy. We compare against representative few-step distillation methods,

including CTM, DMD, and iMF. Across these settings, FastOPD enables the compact student to recover up to 84% of the performance of teacher models that are more than $7.3\times$ larger, using only one or two sampling steps. This substantially reduces inference cost while retaining a large fraction of the foundation policy's performance. To the best of our knowledge, FastOPD is the first OPD framework for VLAs that jointly reduces both model size and the number of sampling steps while retaining the capabilities of a large-scale foundation policy.

2 BACKGROUND

Flow-based Models. Diffusion and flow-based models (Song et al., 2021; Ho et al., 2020; Esser et al., 2024; Lipman et al., 2023) generate high-quality samples through iterative denoising. Let $p_1 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a Gaussian prior and p_0 the context-conditioned data distribution over actions. Flow matching (Lipman et al., 2023; Esser et al., 2024) defines a stochastic interpolant $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$, with $\boldsymbol{x}_0 \sim p_0$, $\boldsymbol{x}_1 \sim p_1$, $t \in [0,1]$, and learns the marginal velocity field $\boldsymbol{v}(\boldsymbol{x}, t) := \mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{x}_0 \mid \boldsymbol{x}_t = \boldsymbol{x}]$ via

$$\mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}\left[\|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|_2^2\right]. \tag{1}$$

Sampling integrates the probability-flow ODE $\dot{\boldsymbol{x}}_t = \boldsymbol{v}(\boldsymbol{x}_t, t)$ from $t = 1$ to $t = 0$, for example, with Euler steps, $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{t_{i-1}} + \boldsymbol{v}(\boldsymbol{x}_{t_{i-1}}, t_{i-1})\,\Delta t$ where $\Delta t := t_i - t_{i-1}$. While these models capture high-quality multimodal action distributions, accurate integration typically requires many denoising steps, motivating few-step distillation for faster inference.

Vision-Language-Action Models. Vision-Language-Action (VLA) models (Collaboration, 2024; Black et al., 2025b;a; Bjorck et al., 2025; Wu et al., 2026; Liu et al., 2024; Octo Model Team et al., 2024) couple a pretrained vision-language backbone with an action decoding head to map an observation and a language instruction to a distribution over robot actions, most commonly via a flow or diffusion action expert (Chi et al., 2023; Black et al., 2025b) trained with flow matching loss. Recent foundation VLA models illustrate the scale required for strong multi-task generalization, typically exceeding 2B parameters and trained on over 10,000 hours of multimodal robot data spanning multiple embodiments (Collaboration, 2024; Black et al., 2025b;a; Bjorck et al., 2025; Wu et al., 2026; Fang et al., 2026a), but remain expensive to deploy. Several works (Octo Model Team et al., 2024; Shukor et al., 2025; Wen et al., 2025; Reuss et al., 2025) instead train smaller VLAs; SmolVLA (Shukor et al., 2025), for example, reduces model size to 450 million parameters by using small VLM and layer skipping. However, it still requires $N{=}10$ denoising steps per action chunk and does not benefit from the capabilities of larger foundation policies.

Flow Map Models. A wide range of few-step generative methods aim to reduce iterative sampling cost. Distillation-based approaches (Sabour et al., 2025; Boffi et al., 2024), including Consistency Models (CM) (Song et al., 2023), and Consistency Trajectory Models (CTM) (Kim et al., 2024), distill a pre-trained multi-step teacher into a few-step student, while score-distillation methods directly match teacher and student distributions (Yin et al., 2024b;a). Distillation-free approaches, such as MeanFlow (MF) (Geng et al., 2025) and Shortcut Models (Frans et al., 2025), instead train from scratch using self-consistency constraints. A flow map model $\boldsymbol{f}_\theta(\boldsymbol{x}, s, t)$ parameterizes the state-to-state transition over an arbitrary interval $[s,t] \subset [0,1]$, so that a single jump $\boldsymbol{f}_\theta(\boldsymbol{x}_s, s, t) = \boldsymbol{x}_s + (t-s)\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t)$ at the interpolant $\boldsymbol{x}_s = (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$ replaces an entire sequence of Euler substeps. For example, Frans et al. (2025) combines flow matching loss with the self-consistency constraint at the midpoint time $m = (s+t)/2$:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,t,\boldsymbol{x}_1,\boldsymbol{x}_0}\left[||\boldsymbol{v}_\theta(\boldsymbol{x}_s, s) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)||_2^2 + ||\boldsymbol{f}_\theta(\boldsymbol{x}_s, s, t) - \boldsymbol{f}_{\theta^-}(\boldsymbol{f}_{\theta^-}(\boldsymbol{x}_s, s, m), m, t)||_2^2\right] \tag{2}$$

The first term enforces standard flow matching, while the second encourages the direct transition from s to t to agree with two shorter transitions composed through m. In robot manipulation, these ideas have been adapted to accelerate diffusion- and flow-based policies such as Diffusion Policy (Chi et al., 2023). Consistency Policy (CP) (Prasad et al., 2024) adapts CTM (Kim et al., 2024); OneDP (Wang et al., 2025) and SDM Policy (Jia et al., 2024) use distribution matching for one-step action generation; and recent methods learn average velocity fields (Geng et al., 2025; 2026; Sheng et al., 2026; Guo et al., 2026) or shortcut objectives (Frans et al., 2025; Luan et al., 2026; Yan et al., 2025; Li et al., 2026b) regularized by flow matching. For instance, SnapFlow (Luan et al., 2026) constructs its target from two Euler steps but trains only on fixed endpoints, while OneFP (Li et al., 2026b) derives its target from nested sub-intervals with a self-guided regularizer. Despite these

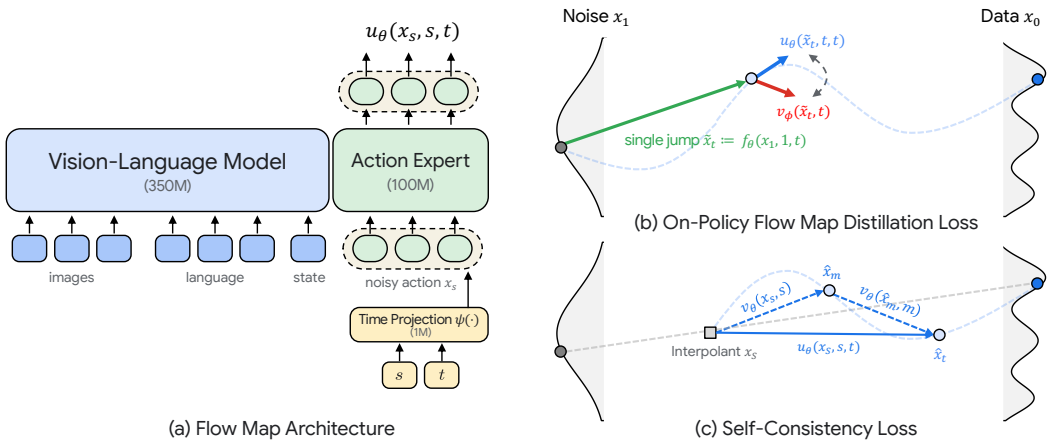


Figure 3: **Overview of FastOPD.** (a) Flow Map Architecture: a frozen vision-language backbone with a lightweight action expert. A time projection layer $\psi(\cdot)$ fuses (s, t) to condition the action expert on the two-time velocity $\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t)$. (b) On-Policy Flow Map Distillation Loss: at a single on-policy jump $\tilde{\boldsymbol{x}}_t = \boldsymbol{f}_\theta(\boldsymbol{x}_1, 1, t)$, the student's diagonal velocity is matched to the teacher's. (c) Self-Consistency Loss: a direct transition $\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t)$ is trained to agree with two steps through the midpoint m.

advances, few-step policies often suffer from training instability and degraded quality as the number of denoising steps decreases. This trade-off is particularly critical in robot manipulation, where low latency must be achieved without sacrificing the precision and stability required for physical execution, and still remains insufficiently addressed (Shukor et al., 2025).

On-policy Distillation. Given a pretrained teacher diffusion model, on-policy distillation (OPD) supervises the student to match the teacher's one-step transition kernel $p_T(\cdot \mid \boldsymbol{x}_{t_i})$ along the student's own rollout $\boldsymbol{x}_{0:N} \sim p_{S,\theta}$, by minimizing the reverse KL objective (Li et al., 2026a). Since the student and teacher share the same diffusion coefficient, resulting in identical covariance, its per-step KL reduces to a closed-form squared distance between the means, $\mathrm{KL}(p_S \| p_T) \propto \|\mu_S(\boldsymbol{x}_{t_i}; \theta) - \mu_T(\boldsymbol{x}_{t_i})\|_2^2$. Despite providing dense supervision, this objective necessitates evaluating the teacher at every single denoising step, resulting in computational inefficiency. FastOPD circumvents this training bottleneck by supervising the student at a single on-policy state per trajectory. Formulated within a few-step flow map regime, the distilled policy is naturally equipped for real-time robot control.

3 FastOPD: Fast On-Policy Distillation

Overview. Our goal is to efficiently distill a large pretrained teacher $\boldsymbol{v}_\phi$ into a compact flow map student $\boldsymbol{u}_\theta$, while achieving performance on par with the teacher. Specifically, we parameterize the flow map as $\boldsymbol{f}_\theta(\boldsymbol{x}, s, t) = \boldsymbol{x} + (t-s)\boldsymbol{u}_\theta(\boldsymbol{x}, s, t)$, which satisfies the boundary condition $\boldsymbol{f}_\theta(\boldsymbol{x}, s, s) = \boldsymbol{x}$. We denote its diagonal by $\boldsymbol{v}_\theta(\boldsymbol{x}, t) := \boldsymbol{u}_\theta(\boldsymbol{x}, t, t)$, which corresponds to the instantaneous velocity. As illustrated in Fig. 3, we embed s and t independently via sinusoidal positional encoding and fuse them through an additional projection layer $\psi(\cdot)$. Here, we omit the condition c for brevity.

The key idea of FastOPD is to match the teacher only at sampled on-policy states and propagate this local velocity supervision to finite-step flow maps through self-consistency. Accordingly, FastOPD jointly optimizes on-policy velocity matching and finite-interval self-consistency:

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}\Big[\big\|\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t) - \tfrac{1}{2}\big(\mathrm{sg}[\boldsymbol{v}_\theta(\boldsymbol{x}_s, s)] + \mathrm{sg}[\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m, m)]\big)\big\|^2\Big]}_{\text{finite-interval self-consistency}} + \lambda \underbrace{\mathbb{E}\Big[\big\|\boldsymbol{v}_\theta(\tilde{\boldsymbol{x}}_t, t) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t, t)\big\|^2\Big]}_{\text{on-policy flow map distillation}}. \tag{3}$$

Here, $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, and $\tilde{\boldsymbol{x}}_t$ denotes a sampled on-policy state, while $\hat{\boldsymbol{x}}_m$ denotes the midpoint state induced by the student. We next describe how the two terms implement local teacher matching and its propagation to finite-step flow maps.

On-policy Flow Map Distillation. We first consider how to efficiently transfer the teacher dynamics to the student. Conventional flow-based OPD evaluates the teacher at every denoising step along the student trajectory, resulting in substantial computational overhead. Instead, we supervise the student at a single sampled on-policy state per trajectory (see Fig. 3(b)). Specifically, we obtain the state with a single flow-map transition from noise, $\tilde{\boldsymbol{x}}_t = \mathrm{sg}\big[\boldsymbol{f}_\theta(\boldsymbol{x}_1, 1, t)\big]$, which induces the on-policy marginal $\tilde{\boldsymbol{x}}_t \sim p_\theta$. We then match the teacher and student transitions only at this sampled timestep t. Let $p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t)$ and $p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t)$ denote the corresponding one-step transition kernels of the student and teacher, respectively. We formulate the *single-state transition matching* objective as

$$\mathbb{E}_{t\sim p(t)}\mathbb{E}_{\tilde{\boldsymbol{x}}_t\sim p_\theta}\left[\mathrm{KL}\left(p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t)\|p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t)\right)\right]. \tag{4}$$

For a flow map, the instantaneous velocity on the diagonal coincides with the underlying velocity field under the tangent condition (Boffi et al., 2025) [1]. Using this identity, the reverse KL in Eq. (4) can be expressed directly in terms of the student and teacher velocity fields, as shown in the following proposition.

Proposition 1 (On-policy Flow Map Distillation)**.** *Let $\sigma_t > 0$ be a diffusion coefficient. For any $t \in (0, 1)$ and step $\Delta t < 0$, the Euler–Maruyama kernels induced by $\boldsymbol{u}_\theta(\cdot, t, t)$ and $\boldsymbol{v}_\phi(\cdot, t)$ satisfy*

$$D_{\mathrm{KL}}\left(p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t) \,\|\, p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t)\right) = w_\sigma(t, \Delta t)\,\|\boldsymbol{u}_\theta(\tilde{\boldsymbol{x}}_t, t, t) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t, t)\|_2^2, \tag{5}$$

where $w_\sigma(t, \Delta t) = \frac{|\Delta t|}{2\sigma_t^2}\left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)^2$.

Proposition 1 reduces the single-state transition matching objective to a weighted velocity regression. We therefore define the practical on-policy flow map distillation (OPFD) objective as

$$\mathcal{L}_{\mathrm{OPFD}}(\theta) = \mathbb{E}_{c,t\sim p(t)}\left[\mathbb{E}_{\tilde{\boldsymbol{x}}_t\sim p_\theta}\left[\|\boldsymbol{v}_\theta(\tilde{\boldsymbol{x}}_t, t) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t, t)\|_2^2\right]\right]. \tag{6}$$

Eq. (6) can be viewed as the pointwise version of the original distribution matching in Eq. (4), since it is characterized by a deterministic flow ODE. We set the weighting function $w_\sigma(t, \Delta t)$ to 1 in practice, following Li et al. (2026a). Full derivation is provided in Appendix A.

Self-consistency Property. OPFD transfers the teacher velocity to the student's diagonal flow map $\boldsymbol{u}_\theta(\boldsymbol{x}, t, t)$, but our goal is to learn a flow map over arbitrary finite interval (s, t) at the same time. We therefore propagate this local teacher supervision to finite-step transitions through self-consistency (see Fig. 3(c)). Following Shortcut Models (Frans et al., 2025), a direct transition from s to t should agree with two consecutive shorter transitions through the midpoint $m = (s + t)/2$. Unlike the original shortcut formulation, we construct the target using the *instantaneous* velocities at the diagonals of the two sub-intervals, $\boldsymbol{v}_\theta(\boldsymbol{x}_s, s)$ and $v_\theta(\hat{x}_m, m)$, as follows:

$$\mathcal{L}_{\mathrm{SC}}(\theta) = \mathbb{E}_{c,s,t,\boldsymbol{x}}\left[\left\|\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t) - \frac{1}{2}\left[\mathrm{sg}\big[\boldsymbol{v}_\theta(\boldsymbol{x}_s, s)\big] + \mathrm{sg}\big[\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m, m)\big]\right]\right\|^2\right] \tag{7}$$

where $\hat{\boldsymbol{x}}_m := \boldsymbol{x}_s + (m - s)\boldsymbol{v}_\theta(\boldsymbol{x}_s, s)$.

The resulting target is anchored to teacher-supervised quantities at every interval. Note that the diagonal terms coincide with the quantities directly supervised by OPFD. Accordingly, we define the training objective as

$$\mathcal{L}_{\text{FastOPD}} = \mathcal{L}_{\mathrm{SC}} + \lambda\mathcal{L}_{\mathrm{OPFD}}. \tag{8}$$

In the following theorem, we demonstrate that the student implicitly recovers the teacher-induced flow map. Let $\boldsymbol{v}_\phi(\boldsymbol{x}, t)$ be the teacher velocity field, and we define the teacher-induced ideal shortcut as

$$\boldsymbol{u}_\phi(\boldsymbol{x}, s, t) = \tfrac{1}{2}\big[\boldsymbol{v}_\phi(\boldsymbol{x}, s) + \boldsymbol{v}_\phi(\bar{\boldsymbol{x}}_m, m)\big], \quad \bar{\boldsymbol{x}}_m = \boldsymbol{x} + (m - s)\,\boldsymbol{v}_\phi(\boldsymbol{x}, s), \tag{9}$$

with $\boldsymbol{u}_\phi(\boldsymbol{x}, t, t) = \boldsymbol{v}_\phi(\boldsymbol{x}, t)$. We then define the corresponding ideal flow map as $\boldsymbol{f}_\phi(\boldsymbol{x}, s, t) = \boldsymbol{x} + (t - s)\boldsymbol{u}_\phi(\boldsymbol{x}, s, t)$.

Proposition 2. *Let q_θ and ρ_θ be the sampling distributions of $\mathcal{L}_{SC}$ and $\mathcal{L}_{OPFD}$. Assume that $\boldsymbol{v}_\phi(\cdot, t)$ is L-Lipschitz and that $(s, \boldsymbol{x}_s)$ and $(m, \hat{\boldsymbol{x}}_m)$ under q_θ have density ratio at most κ with respect to ρ_θ. Then,*

$$D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi) \le C_\lambda\mathcal{L}_{\textit{FastOPD}}, \tag{10}$$

where $D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi) := \mathbb{E}_{(s,t,\boldsymbol{x})\sim q_\theta}\|\boldsymbol{u}_\theta(\boldsymbol{x}, s, t) - \boldsymbol{u}_\phi(\boldsymbol{x}, s, t)\|_2^2$ *and* $C_\lambda = \max\left\{4,\ \frac{\kappa(2+L^2/4)}{\lambda}\right\}$.

[1] $\lim_{s\to t}\partial_t\boldsymbol{f}_\theta(\boldsymbol{x}, s, t) = \boldsymbol{u}_\theta(\boldsymbol{x}, t, t) = \boldsymbol{v}_\theta(\boldsymbol{x}, t)$

Algorithm 1 Training Algorithm for FastOPD

Require: Dataset $\mathcal{D}$; Pretrained teacher $\boldsymbol{v}_\phi$; student flow map $\boldsymbol{u}_\theta$ with time embedding layer $\psi(\cdot)$; OPFD weight λ; Sampling time distribution p

for each training iteration **do**
  $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ \tau \sim p,\ (\boldsymbol{x}_0, c) \sim \mathcal{D}$
  $\tilde{\boldsymbol{x}}_\tau \leftarrow \mathrm{sg}\big[\boldsymbol{f}_\theta(\boldsymbol{x}_1, 1, \tau)\big]$
  $\mathcal{L}_{\text{OPFD}} \leftarrow \|\boldsymbol{v}_\theta(\tilde{\boldsymbol{x}}_\tau, \tau \mid c) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_\tau, \tau \mid c)\|_2^2$ ▷ *single on-policy state supervision*
  $s, t \sim p$ with $s > t$
  $m \leftarrow (s+t)/2,\ \boldsymbol{x}_s \leftarrow (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$
  $\hat{\boldsymbol{x}}_m \leftarrow \boldsymbol{x}_s + (m-s)\,\mathrm{sg}\big[\boldsymbol{v}_\theta(\boldsymbol{x}_s, s \mid c)\big]$
  $\mathcal{L}_{\text{SC}} \leftarrow \big\|\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t \mid c) - \frac{1}{2}\big[\mathrm{sg}[\boldsymbol{v}_\theta(\boldsymbol{x}_s, s \mid c)] + \mathrm{sg}[\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m, m \mid c)]\big]\big\|_2^2$
  $\mathcal{L}_{\text{FastOPD}} \leftarrow \mathcal{L}_{\text{SC}} + \lambda\,\mathcal{L}_{\text{OPFD}}$
  $\theta \leftarrow \theta - \eta\,\nabla_\theta \mathcal{L}_{\text{FastOPD}}$
end for

Thus, OPFD transfers the teacher's instantaneous velocity to the student, while self-consistency propagates this agreement from the diagonal velocity to finite-step shortcut maps.

Theorem 1. *Let p_θ and p_ϕ denote the terminal distributions induced by sampling with $\boldsymbol{f}_\theta$ and $\boldsymbol{f}_\phi$, respectively, from the same initial noise on a fixed sampling schedule $1 = t_0 > t_1 > \cdots > t_K = 0$, and let $\boldsymbol{x}^\theta_{t_k}$ denote the intermediate states of the student. Suppose that the assumptions of Proposition 2 hold and there exists finite $\kappa_{\mathrm{roll}} > 0$ such that, for every k,*

$$D_k := \mathbb{E}\big\|\boldsymbol{u}_\theta(\boldsymbol{x}^\theta_{t_k}, t_k, t_{k+1}) - \boldsymbol{u}_\phi(\boldsymbol{x}^\theta_{t_k}, t_k, t_{k+1})\big\|_2^2 \le \kappa_{\mathrm{roll}}\, D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi). \tag{11}$$

Then,

$$W_2(p_\theta, p_\phi) \le C\sqrt{\mathcal{L}_{\textit{FastOPD}}}, \tag{12}$$

where $C > 0$ depends only on L, κ, κ_{roll}, λ, and the sampling schedule.

Consequently, $W_2(p_\theta, p_\phi) \to 0$ as $\mathcal{L}_{\text{FastOPD}} \to 0$. Since $\boldsymbol{u}_\phi$ is built entirely from the pretrained teacher's velocity field, p_ϕ is the few-step distribution induced by the teacher-induced flow map on the same sampling schedule. The theorem therefore shows that FastOPD allows the compact student to implicitly recover this distribution without explicitly constructing a shortcut teacher. The detailed algorithm for FastOPD is presented in Algorithm 1.

4 Experiments

Our experiments span two multitask simulation benchmarks (Sec. 4.1) and one real-robot task (Sec. 4.2). We freeze the vision–language backbone and fine-tune only the action expert and projections with the additional time projection layer described in Sec. 3. We evaluate each method using 1, 2, 4, and 10 denoising steps and compare it against the teacher and the initial student with 50 trials per task. Full experimental details are provided in Appendix B.

4.1 Simulation Experiments

Benchmarks. We evaluate FastOPD on LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2025). LIBERO evaluates single-arm manipulation across four standard evaluation suites—Spatial, Object, Goal, and Long—comprising 40 tasks in total. RoboTwin 2.0 (Chen et al., 2025) comprises 50 challenging bimanual manipulation tasks, with 2,500 demonstrations collected in clean scenes and 25,000 demonstrations collected under heavy scene randomization. We report results in both clean and randomized settings on unseen environments.

Baselines. For simulation experiments, we compare our method with three representative approaches to few-step generation: Consistency Trajectory Model (CTM) (Kim et al., 2024), Distribution Matching Distillation (DMD) (Yin et al., 2024b;a), and improved Mean Flow (iMF) (Geng et al., 2026). Since iMF was originally formulated for training from scratch, we adapt it to the distillation setting by replacing its self-referential instantaneous velocity target with the teacher velocity. Except for DMD, a one-step generator, all baselines are designed as the flow-map architecture.

Table 1: Success rates (%) on LIBERO across spatial, object, goal, and long-horizon tasks. †: Results reported in the original papers.

Method	Param.	NFE	Spat.	Obj.	Goal	Long	Avg.
Diffusion/Flow-based Policies							
DP†	–	100	78.3	92.5	68.3	50.5	72.4
Octo-Base†	93M	10	78.9	85.7	84.6	51.1	75.1
π_0†	3.3B	10	97.4	98.4	97.6	93.0	96.6
Teacher & Base Student Models							
$\pi_{0.5}$	3.3B	10	98.2	99.4	97.4	94.8	97.5
SmolVLA	450M	1	74.6	73.2	77.4	51.2	69.1
		2	76.8	74.4	79.8	54.4	71.4
		4	79.0	74.8	79.2	58.0	72.8
		10	77.2	75.0	79.4	54.6	71.6
Few-step Distillation							
DMD	450M	1	82.4	87.2	75.6	47.2	73.1
CTM	451M	1	78.0	86.6	70.8	41.4	69.2
		2	75.4	84.2	70.4	43.4	68.4
		4	68.4	77.0	71.2	45.0	65.4
		10	72.0	81.6	72.4	44.4	67.6
iMF	451M	1	70.8	78.6	75.6	30.6	63.9
		2	76.0	90.6	75.6	46.4	72.2
		4	68.2	84.8	67.0	38.0	64.5
		10	65.2	83.6	66.6	35.2	62.7
Ours	451M	1	84.2	94.2	82.6	48.2	**77.3**
		2	83.6	96.8	87.2	59.6	**81.8**
		4	83.0	95.0	85.6	57.4	**80.3**
		10	80.4	94.0	84.4	57.2	**79.0**

Table 2: Success rates (%) on RoboTwin 2.0 under unseen clean and randomized settings. †: Results reported in the original papers.

Method	Param.	NFE	Clean	Rand.	Avg.
Diffusion/Flow-based Policies					
DP†	0.46B	10	28.0	0.6	14.3
RDT†	1.2B	10	34.5	13.7	24.1
π_0†	3.3B	10	46.4	16.3	31.4
$\pi_{0.5}$	3.3B	10	70.7	46.0	58.4
Teacher & Base Student Models					
LingBot-VLA	4.2B	10	86.8	84.9	85.8
Fast-WAM	6B	10	91.9	91.8	91.9
SmolVLA	450M	1	35.2	35.4	35.3
		2	55.0	56.1	55.6
		4	59.8	61.0	**60.4**
		10	59.4	61.3	**60.4**
Few-step Distillation					
DMD	450M	1	32.5	32.1	32.3
CTM	451M	1	15.5	15.6	15.6
		2	4.8	15.8	10.3
		4	18.7	16.0	17.3
		10	36.1	34.5	35.3
iMF	451M	1	43.2	42.4	42.8
		2	51.6	51.1	51.4
		4	50.8	51.0	50.9
		10	50.6	51.4	51.0
Ours (LingBot-VLA)	451M	1	51.5	50.9	**51.2**
		2	57.1	58.3	**57.7**
		4	58.6	61.2	59.9
		10	56.9	59.6	58.2
Ours (Fast-WAM)	451M	1	51.1	49.8	50.5
		2	55.5	57.2	56.3
		4	58.5	59.2	58.8
		10	58.3	59.6	58.9

Table 3: End-to-end inference latency (ms) on LIBERO and RoboTwin 2.0.

LIBERO					RoboTwin 2.0				
Model	**NFE=1**	**2**	**4**	**10**	**Model**	**NFE=1**	**2**	**4**	**10**
$\pi_{0.5}$ (teacher)	155	171	203	301	LingBot-VLA (teacher)	151	197	291	571
					Fast-WAM (teacher)	147	186	264	503
SmolVLA	49	65	97	193	SmolVLA	56	72	103	199
Ours	50	66	98	193	**Ours**	57	74	106	202

Teacher and Student Models. We use SmolVLA (Shukor et al., 2025) as the student model on both benchmarks, paired with $\pi_{0.5}$ (Black et al., 2025a) on LIBERO, LingBot-VLA (Wu et al., 2026) and Fast-WAM (Yuan et al., 2026) on RoboTwin 2.0. All models are initialized from public benchmark-specific checkpoints.

LIBERO. In Tab. 1 and Tab. 3, we report the average success rates and inference latency on the LIBERO benchmark. Our method outperforms all few-step baselines and the initial student model at every denoising step in average success rate. As shown in Fig. 4, we observe a consistent performance improvement across all denoising steps, including at 10 steps. FastOPD further achieves the best success rates of 81.8% with only 2 sampling steps. In particular, our method shows better performance than the 10-step SmolVLA baseline with only a single step, while reducing inference latency from 193 ms to 50 ms. Together, the results demonstrate that FastOPD substantially improves the accuracy–latency trade-off of the compact student. We examine the contributions of each loss component to these accuracy gains in Sec. 4.3.

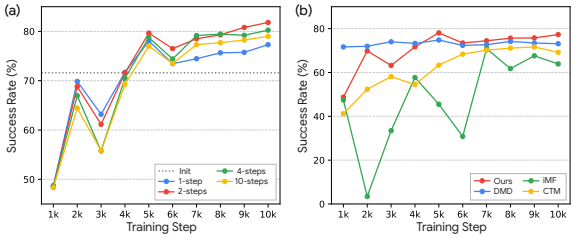


Figure 4: Success rate (%) on LIBERO during training: (a) FastOPD at different sampling steps, (b) comparison with baselines at 1 step.

RoboTwin 2.0. Tab. 2 and Tab. 3 summarize the results. Notably, our method consistently achieves performance gain in the few-step regime, improving the single-step success rate of the base student from 35.3% to 51.2%, and outperforms all few-step distillation baselines at every sampling step. It does so at a latency comparable to SmolVLA, while being $10.0\times$ faster than the LingBot-VLA teacher. At 4 and 10 steps, FastOPD performs slightly below the base student, indicating that its benefit lies mainly in the few-step regime where the base student degrades.

Distilling a World Action Model. Recent World Action Models (WAMs) build on pretrained video generation models and have emerged as an alternative to foundation VLAs (Kim et al., 2026; Ye et al., 2026; Yuan et al., 2026). To examine whether FastOPD extends beyond VLA teachers, we distill Fast-WAM (Yuan et al., 2026), a 6B-parameter video-based teacher, into the same 451M SmolVLA student with an identical training recipe. As shown in Tab. 2, the student distilled from Fast-WAM achieves a 50.5% average success rate at a single denoising step, a 15.2% percentage points over the initial SmolVLA baseline (35.3%). Furthermore, it reaches 56.3% at two steps, successfully transferring knowledge despite the substantial differences in architecture and pretraining between the teacher and the student. This demonstrates that FastOPD is a versatile approach that applies not only to VLAs but also to WAMs, since it only requires the teacher's velocity field on the student's own action trajectories, regardless of how the teacher is built.

4.2 Real-World Experiments

To validate its real-world applicability, we further design a real-world robot scenario, `pnp-plate`, in which the robot picks up the ball and places it on the plate. Experiments use the right arm of a bimanual YAM robot with a parallel-jaw gripper and top-view and wrist-mounted cameras that provide RGB observations. We train MolmoAct2 (Fang et al., 2026a), an action reasoning model that is suitable for real-world deployment, on our collected demonstrations for 14,000 iterations and use the resulting policy as the teacher. We initialize the student from SmolVLA, warm it up on the collected demonstrations with the standard flow matching objective (i.e., behavior cloning) for 10,000 iterations, and subsequently distill the teacher into the student using FastOPD. Full experimental details are provided in Appendix B.5.

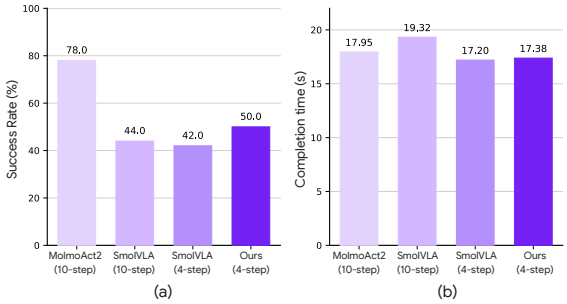


Figure 5: Real-world results on the `pnp-plate` task. (a) Success rate (%). (b) Task completion time (s), averaged over successful episodes.

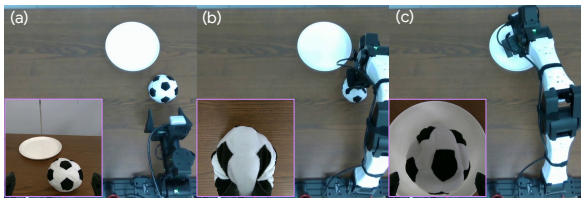


Figure 6: FastOPD on `pnp-plate`: (a) start, (b) transport, and (c) placement. Each panel shows the top view with the wrist view inset (lower left).

Fig. 5 and Fig. 6 summarize the results. With four denoising steps, FastOPD achieves a 50% success rate, improving by 8 percentage points over the 42% of the base SmolVLA at the same number of steps. Notably, FastOPD shows faster and more accurate performance than the 10-step base student, improving the success rate from 44% to 50% while cutting the execution time from 19.32 s to 17.38 s. These results demonstrate that FastOPD can transfer the capability of a larger teacher to a compact student deployed on a physical robot with few steps.

4.3 Ablation Study

Training efficiency. FastOPD reduces training overhead by supervising the student at a single sampled on-policy state rather than across all denoising steps. As illustrated in Fig. 1, standard OPD (Li et al., 2026a) takes 9.51 hours per 1,000 training iterations while FastOPD requires only 1.43 hours, achieving a $5.7\times$ speedup. This translates directly into accelerated convergence (Fig. 1). FastOPD achieves a 78.2% 1-step success rate in just 10 hours, whereas conventional OPD requires

Table 4: Student initialization.

Method	1	2	4	10
FM init.	69.1	71.4	72.8	71.6
+ FastOPD	77.3	81.8	80.3	79.0
OPD init.	72.7	78.2	44.2	78.6
+ FastOPD	**80.5**	**83.0**	**82.2**	**81.7**

Table 5: Loss components.

Method	1	2	4	10
Base	69.1	71.4	72.8	71.6
SC only	14.5	15.7	17.8	16.8
OPFD only	25.0	52.6	64.0	74.7
FastOPD	**77.3**	**81.8**	**80.3**	**79.0**

Table 6: Effect of λ.

λ	1	2	4	10
1	77.9	82.0	83.6	82.1
0.5	79.3	82.9	**84.2**	**84.1**
0.1	**80.5**	**83.0**	82.2	81.7
0.01	53.0	52.6	52.9	52.0

roughly 57 hours to reach a similar level (79.1%). Consequently, FastOPD matches the performance of conventional OPD at a fraction of the computational cost.

Student Initialization. We study whether FastOPD can also start from a student that is initialized in a fully on-policy manner. To do so, we initialize the student using 5,000 iterations of conventional per-step OPD (Li et al., 2026a) instead of flow-matching behavior cloning. As shown in Tab. 4, the OPD-initialized student degrades severely at 4 steps (44.2%), likely due to the fixed 10-step discretization in OPD, whereas FastOPD recovers this case and improves all denoising steps. This indicates that OPD is a viable alternative to flow-matching-based behavior cloning for the initialization, although it incurs a substantially higher training cost, as discussed above (Fig. 1). It also shows that FastOPD is not tied to a specific initialization and can be applied on top of students distilled by other methods.

Loss components. Tab. 5 shows that the two objectives are complementary. Without teacher anchoring, SC alone collapses below 18% and OPFD alone fails to capture off-diagonal flow dynamics, degrading few-step generation. Their combination reaches 81.8% with two steps. While Luan et al. (2026) also anchors a fixed two-step Euler target with flow matching, FastOPD anchors to the teacher via OPFD and enforces consistency over arbitrary intervals, yielding an any-step flow map that inherits the teacher's dynamics (see Appendix C).

Hyperparameters. The proposed method introduces a single hyperparameter λ, which controls the strength of the OPFD term. To study its effect, we sweep the OPFD weight $\lambda \in \{1, 0.5, 0.1, 0.01\}$ on the OPD-initialized student (Tab. 6). Performance collapses at $\lambda = 0.01$, indicating that sufficient OPFD supervision is needed to anchor the self-consistency term, and remains stable within $[0.1, 0.5]$ with a mild trade-off between few- and multi-step performance.

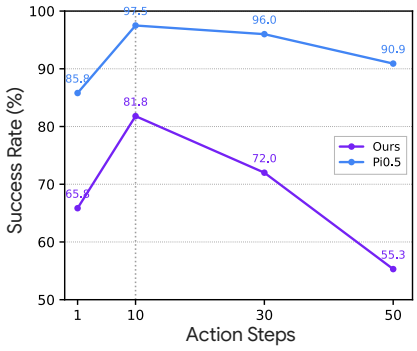


Figure 7: Ablation on action execution steps on LIBERO (NFE=2 for Ours and NFE=10 for $\pi_{0.5}$).

Action execution steps. The policy executes the first K actions of each predicted chunk before replanning, trading responsiveness for inference overhead. Short execution horizons improve responsiveness through more frequent replanning, while longer horizons reduce inference overhead at the cost of extended open-loop execution. As shown in Fig. 7, we observe that both the teacher $\pi_{0.5}$ and the student perform best at $K = 10$, suggesting that the student inherits the teacher's preferred replanning frequency.

5 Conclusion

In this paper, we present FastOPD, an efficient on-policy distillation framework that transfers large-scale VLAs into lightweight student models to maximize computational efficiency without compromising performance. Specifically, FastOPD streamlines the iterative denoising process by anchoring a single on-policy state to the teacher, coupled with a self-consistency objective for finite-interval transitions. In simulation and real-robot experiments across diverse VLA and WAM teachers, FastOPD improves over the base student in the few-step regime and outperforms existing few-step distillation methods, while substantially reducing training and inference cost. We hope this framework facilitates scalable, real-time deployment of foundation policies on physical robots.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing and checking the derivations in Appendix A for errors. We have not used generative AI tools for developing theoretical models, formulate mathematical claims, or design the research methodology, and the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools to assist in implementing the baselines. We have reviewed all AI-assisted work. All proofs elaborated by the AI were reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or personally identifiable information. Demonstrations for real-world experiments were collected by the authors via teleoperation of a YAM robot arm in a controlled laboratory setting using everyday objects (balls, plates, and boxes). We do not foresee direct negative societal impacts beyond the general safety considerations of deploying autonomous robots in physical environments.

REPRODUCIBILITY STATEMENT

We provide the full training procedure in Algorithm 1 and complete proofs of all theoretical results, including their assumptions, in Appendix A. All teacher and student models are initialized from publicly available checkpoints listed in Tab. 7. Training and evaluation configurations are given in Tab. 8, hardware in Tab. 9, and evaluation protocols, including latency measurement, in Appendix B. Implementation details of all baselines are described in Appendix D, and the real-world robot setup, dataset statistics, and evaluation criteria in Appendix B.5. All experiments are implemented in the open-source LeRobot framework.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025b.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38: 37559–37595, 2025.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in Pytorch. `https://github.com/huggingface/lerobot`, 2024.

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

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

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR, 2024.

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

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, et al. Flow-OPD: On-policy distillation for flow matching models. *arXiv preprint arXiv:2605.08063*, 2026b.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026.

Yanzhao Guo, Wenkai Chen, and Jianwei Zhang. ReactVLA: Fast and lightweight reactive robot manipulation via improved mean flow action generation. *arXiv preprint arXiv:2606.14255*, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Bofang Jia, Pengxiang Ding, Can Cui, Mingyang Sun, Pengfang Qian, Siteng Huang, Zhaoxin Fan, and Donglin Wang. Score and distribution matching policy: Advanced accelerated visuomotor policies via matched distillation. *arXiv preprint arXiv:2412.09265*, 2024.

Dongjun Kim, Chieh-Hsin Lai, WeiHsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. DiffusionOPD: A unified perspective of on-policy distillation in diffusion models. In *SIGGRAPH Asia 2026 Conference Papers*, 2026a.

Shaolong Li, Lichao Sun, and Yongchao Chen. One-step flow policy: Self-distillation for fast visuomotor policies. *arXiv preprint arXiv:2603.12480*, 2026b.

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.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. SnapFlow: One-step action generation for flow-matching VLAs via progressive self-distillation. *arXiv preprint arXiv:2604.05656*, 2026.

Amir M Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab Yasamani Ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, et al. A comprehensive survey on knowledge distillation. *arXiv preprint arXiv:2503.12067*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Conference on Robot Learning*, 2025.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. MP1: MeanFlow tames policy learning in 1-step for robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18532–18539, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

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.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252, 2023.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In *International Conference on Machine Learning*, 2025.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, et al. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Shuailei Ma, He Sun, Yong Wang, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Shuai Zhou, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Qian Zhu, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692v1*, 2026.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, and Dieter Fox. ManiFlow: A general robot manipulation policy via consistency flow training. In *Conference on Robot Learning (CoRL)*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623. IEEE, 2024b.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. VLA-OPD: Bridging offline SFT and online RL for vision-language-action models via on-policy distillation. *arXiv preprint arXiv:2603.26666*, 2026.

A PROOFS

Proposition 1 (On-policy Flow Map Distillation)**.** *Let $\sigma_t > 0$ be a diffusion coefficient. For any $t \in (0,1)$ and step $\Delta t < 0$, the Euler–Maruyama kernels induced by $\boldsymbol{u}_\theta(\cdot,t,t)$ and $\boldsymbol{v}_\phi(\cdot,t)$ satisfy*

$$D_{\mathrm{KL}}\left(p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t) \,\|\, p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t)\right) = w_\sigma(t,\Delta t)\, \|\boldsymbol{u}_\theta(\tilde{\boldsymbol{x}}_t,t,t) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t,t)\|_2^2\,, \tag{5}$$

where $w_\sigma(t,\Delta t) = \frac{|\Delta t|}{2\sigma_t^2}\left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)^2$.

Proof. For the linear interpolant

$$\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1, \qquad \boldsymbol{x}_0 \sim \mathcal{D}, \qquad \boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0},\mathbf{I}),$$

the score and the corresponding instantaneous velocity field satisfy

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) = -\frac{1}{t}\left(\boldsymbol{x} + (1-t)\boldsymbol{v}(\boldsymbol{x},t)\right). \tag{13}$$

Following the flow-to-SDE conversion of Liu et al. (2025), for any diffusion coefficient $\sigma_t > 0$, the reverse-time SDE that share the same marginal p_t of PF-ODE is defined as

$$d\boldsymbol{x}_t = \left(\boldsymbol{v}(\boldsymbol{x}_t,t) - \frac{\sigma_t^2}{2}\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)\right) dt + \sigma_t\, d\bar{\mathbf{w}}_t. \tag{14}$$

Substituting Eq. (13) into Eq. (14) gives

$$d\boldsymbol{x}_t = \left[\left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)\boldsymbol{v}(\boldsymbol{x}_t, t) + \frac{\sigma_t^2}{2t}\boldsymbol{x}_t\right] dt + \sigma_t\, d\bar{\mathbf{w}}_t. \tag{15}$$

A single Euler–Maruyama step of size Δt from $\tilde{\boldsymbol{x}}_t$ therefore defines the Gaussian transition kernel

$$p_{\boldsymbol{v}}(\boldsymbol{x}_{t+\Delta t} \mid \tilde{\boldsymbol{x}}_t) = \mathcal{N}\left(\mu_{\boldsymbol{v}}, \sigma_t^2|\Delta t|\,\mathbf{I}_d\right), \tag{16}$$

where

$$\mu_{\boldsymbol{v}} = \left(1 + \frac{\sigma_t^2}{2t}\Delta t\right)\tilde{\boldsymbol{x}}_t + \left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)\boldsymbol{v}(\tilde{\boldsymbol{x}}_t, t)\Delta t. \tag{17}$$

For the flow-map parameterization $\boldsymbol{f}_\theta(\boldsymbol{x}, s, t) = \boldsymbol{x} + (t-s)\boldsymbol{u}_\theta(\boldsymbol{x}, s, t)$, its tangent velocity is

$$\partial_\tau \boldsymbol{f}_\theta(\boldsymbol{x}, t, \tau)|_{\tau=t} = \boldsymbol{u}_\theta(\boldsymbol{x}, t, t). \tag{18}$$

We consequently define the student and teacher velocities at $\tilde{\boldsymbol{x}}_t$ by

$$\boldsymbol{v}_S := \boldsymbol{u}_\theta(\tilde{\boldsymbol{x}}_t, t, t), \qquad \boldsymbol{v}_T := \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t, t).$$

Applying Eq. (16) to these two velocities gives

$$p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t) = \mathcal{N}\left(\mu_S, \sigma_t^2|\Delta t|\,\mathbf{I}_d\right), \qquad p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t) = \mathcal{N}\left(\mu_T, \sigma_t^2|\Delta t|\,\mathbf{I}_d\right).$$

Their covariances are identical, and the common $\tilde{\boldsymbol{x}}_t$-dependent term cancels from the difference of their means. Hence,

$$\mu_S - \mu_T = \left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)\Delta t\,(\boldsymbol{v}_S - \boldsymbol{v}_T). \tag{19}$$

Using the KL divergence between Gaussians with common covariance, we obtain

$$D_{\mathrm{KL}}\left(p_\theta(\cdot \mid \tilde{\boldsymbol{x}}_t) \,\|\, p_\phi(\cdot \mid \tilde{\boldsymbol{x}}_t)\right) = \frac{\|\mu_S - \mu_T\|_2^2}{2\sigma_t^2|\Delta t|} \tag{20}$$

$$= \frac{|\Delta t|}{2\sigma_t^2}\left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)^2 \|\boldsymbol{v}_S - \boldsymbol{v}_T\|_2^2 \tag{21}$$

$$= w_\sigma(t, \Delta t)\,\|\boldsymbol{u}_\theta(\tilde{\boldsymbol{x}}_t, t, t) - \boldsymbol{v}_\phi(\tilde{\boldsymbol{x}}_t, t)\|_2^2, \tag{22}$$

where

$$w_\sigma(t, \Delta t) := \frac{|\Delta t|}{2\sigma_t^2}\left(1 + \frac{\sigma_t^2(1-t)}{2t}\right)^2. \tag{23}$$

□

Proposition 2. *Let q_θ and ρ_θ be the sampling distributions of $\mathcal{L}_{SC}$ and $\mathcal{L}_{OPFD}$. Assume that $\boldsymbol{v}_\phi(\cdot, t)$ is L-Lipschitz and that $(s, \boldsymbol{x}_s)$ and $(m, \hat{\boldsymbol{x}}_m)$ under q_θ have density ratio at most κ with respect to ρ_θ. Then,*

$$D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi) \le C_\lambda \mathcal{L}_{\textit{FastOPD}}, \tag{10}$$

where $D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi) := \mathbb{E}_{(s,t,\boldsymbol{x})\sim q_\theta}\|\boldsymbol{u}_\theta(\boldsymbol{x}, s, t) - \boldsymbol{u}_\phi(\boldsymbol{x}, s, t)\|_2^2$ and $C_\lambda = \max\left\{4,\ \frac{\kappa(2+L^2/4)}{\lambda}\right\}$.

Proof. Let $(s, t, \boldsymbol{x}) \sim q_\theta$ with $\boldsymbol{x} = \boldsymbol{x}_s = (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$ and $m = (s+t)/2$. Since $s > t$, we have $|m - s| = (s-t)/2 \le 1/2$.

We define the self-consistency target of the student as

$$T_\theta(\boldsymbol{x}, s, t) = \frac{1}{2}\big[\boldsymbol{v}_\theta(\boldsymbol{x}, s) + \boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m, m)\big], \tag{24}$$

where $\hat{\boldsymbol{x}}_m = \boldsymbol{x} + (m-s)\,\boldsymbol{v}_\theta(\boldsymbol{x}, s)$, and denote the velocity errors at s and m by

$$e_s := \|\boldsymbol{v}_\theta(\boldsymbol{x}, s) - \boldsymbol{v}_\phi(\boldsymbol{x}, s)\|_2, \qquad e_m := \|\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m, m) - \boldsymbol{v}_\phi(\hat{\boldsymbol{x}}_m, m)\|_2. \tag{25}$$

Using the definition of $\boldsymbol{u}_\phi$, we decompose the difference between the student map and the ideal shortcut as

$$\begin{aligned}\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-\boldsymbol{u}_\phi(\boldsymbol{x},s,t) &= \big(\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-T_\theta\big)+\big(T_\theta-\boldsymbol{u}_\phi(\boldsymbol{x},s,t)\big)\\ &= \big(\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-T_\theta\big)+\tfrac{1}{2}\big[\boldsymbol{v}_\theta(\boldsymbol{x},s)-\boldsymbol{v}_\phi(\boldsymbol{x},s)\big]\\ &\quad+\tfrac{1}{2}\big[\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_m,m)-\boldsymbol{v}_\phi(\hat{\boldsymbol{x}}_m,m)\big]+\tfrac{1}{2}\big[\boldsymbol{v}_\phi(\hat{\boldsymbol{x}}_m,m)-\boldsymbol{v}_\phi(\bar{\boldsymbol{x}}_m,m)\big],\end{aligned} \tag{26}$$

where the first term is the self-consistency error, the second and third terms are the velocity errors at s and m, respectively, and the last term is the propagated state error induced by the mismatch between the student midpoint $\hat{\boldsymbol{x}}_m$ and the ideal midpoint $\bar{\boldsymbol{x}}_m$.

By the L-Lipschitzness of the teacher,

$$\|\boldsymbol{v}_\phi(\hat{\boldsymbol{x}}_m,m)-\boldsymbol{v}_\phi(\bar{\boldsymbol{x}}_m,m)\|_2 \le L\|\hat{\boldsymbol{x}}_m-\bar{\boldsymbol{x}}_m\|_2 = L\,|m-s|\,e_s. \tag{27}$$

Therefore,

$$\begin{aligned}\|\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-\boldsymbol{u}_\phi(\boldsymbol{x},s,t)\|_2^2 &\le 4\|\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-T_\theta\|_2^2+e_s^2+e_m^2+L^2(m-s)^2e_s^2\\ &\le 4\|\boldsymbol{u}_\theta(\boldsymbol{x},s,t)-T_\theta\|_2^2+\Big(1+\frac{L^2}{4}\Big)e_s^2+e_m^2,\end{aligned} \tag{28}$$

where the last inequality uses $|m-s|\le 1/2$.

By taking the expectation of Eq. (28) over q_θ, the left-hand side becomes $D_{q_\theta}(\boldsymbol{u}_\theta,\boldsymbol{u}_\phi)$ and the first term on the right-hand side becomes $4\mathcal{L}_{\text{SC}}$. The remaining two terms are velocity errors evaluated at $(s,\boldsymbol{x}_s)$ and $(m,\hat{\boldsymbol{x}}_m)$, whereas $\mathcal{L}_{\text{OPFD}}$ averages the same error over $(t,\tilde{\boldsymbol{x}}_t)\sim\rho_\theta$.

Let $g(t,\boldsymbol{x}) := \|\boldsymbol{v}_\theta(\boldsymbol{x},t)-\boldsymbol{v}_\phi(\boldsymbol{x},t)\|_2^2 \ge 0$, so that $\mathcal{L}_{\text{OPFD}} = \mathbb{E}_{\rho_\theta}[g]$.

Then, for any distribution ν whose density ratio with respect to ρ_θ is at most κ,

$$\mathbb{E}_\nu[g] = \mathbb{E}_{\rho_\theta}\Big[g\,\frac{d\nu}{d\rho_\theta}\Big] \le \kappa\,\mathbb{E}_{\rho_\theta}[g] = \kappa\,\mathcal{L}_{\text{OPFD}}. \tag{29}$$

Applying this with ν the distributions of $(s,\boldsymbol{x}_s)$ and $(m,\hat{\boldsymbol{x}}_m)$, which satisfy this condition by the density-ratio assumption, gives $\mathbb{E}_{q_\theta}[e_s^2]\le\kappa\mathcal{L}_{\text{OPFD}}$ and $\mathbb{E}_{q_\theta}[e_m^2]\le\kappa\mathcal{L}_{\text{OPFD}}$.

Therefore,

$$\begin{aligned}D_{q_\theta}(\boldsymbol{u}_\theta,\boldsymbol{u}_\phi) &\le 4\,\mathcal{L}_{\text{SC}}+\kappa\Big(1+\frac{L^2}{4}\Big)\mathcal{L}_{\text{OPFD}}+\kappa\,\mathcal{L}_{\text{OPFD}}\\ &= 4\,\mathcal{L}_{\text{SC}}+\kappa\Big(2+\frac{L^2}{4}\Big)\mathcal{L}_{\text{OPFD}}\\ &\le \max\Big\{4,\ \frac{\kappa(2+L^2/4)}{\lambda}\Big\}\,\mathcal{L}_{\text{FastOPD}}\\ &= C_\lambda\,\mathcal{L}_{\text{FastOPD}}.\end{aligned} \tag{30}$$

Here, C_λ grows as $\lambda\to 0$, reflecting that the self-consistency term alone cannot anchor the student to the teacher.

□

Theorem 1. *Let p_θ and p_ϕ denote the terminal distributions induced by sampling with $\boldsymbol{f}_\theta$ and $\boldsymbol{f}_\phi$, respectively, from the same initial noise on a fixed sampling schedule $1=t_0>t_1>\cdots>t_K=0$, and let $\boldsymbol{x}^\theta_{t_k}$ denote the intermediate states of the student. Suppose that the assumptions of Proposition 2 hold and there exists finite $\kappa_{\text{roll}}>0$ such that, for every k,*

$$D_k := \mathbb{E}\big\|\boldsymbol{u}_\theta(\boldsymbol{x}^\theta_{t_k},t_k,t_{k+1})-\boldsymbol{u}_\phi(\boldsymbol{x}^\theta_{t_k},t_k,t_{k+1})\big\|_2^2 \le \kappa_{\text{roll}}\,D_{q_\theta}(\boldsymbol{u}_\theta,\boldsymbol{u}_\phi). \tag{11}$$

Then,

$$W_2(p_\theta,p_\phi) \le C\sqrt{\mathcal{L}_{\textit{FastOPD}}}, \tag{12}$$

where $C>0$ depends only on L, κ, κ_{roll}, λ, and the sampling schedule.

Proof. Let the student and the teacher-induced shortcut model start from the same initial noise sample, $\boldsymbol{x}_{t_0}^{\theta} = \bar{\boldsymbol{x}}_{t_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and follow the same sampling schedule

$$1 = t_0 > t_1 > \cdots > t_K = 0 \tag{31}$$

with step sizes $h_k := t_k - t_{k+1} > 0$.

Their rollout states are defined by

$$\boldsymbol{x}_{t_{k+1}}^{\theta} = \boldsymbol{f}_\theta(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1}) \quad \text{and} \quad \bar{\boldsymbol{x}}_{t_{k+1}} = \boldsymbol{f}_\phi(\bar{\boldsymbol{x}}_{t_k}, t_k, t_{k+1}), \tag{32}$$

respectively.

By the definition of $\boldsymbol{u}_\phi$ and $m_k = (t_k + t_{k+1})/2$,

$$\boldsymbol{f}_\phi(\boldsymbol{x}, t_k, t_{k+1}) = \boldsymbol{x} - h_k \boldsymbol{u}_\phi(\boldsymbol{x}, t_k, t_{k+1}) = \bar{\boldsymbol{x}}_{m_k} - \tfrac{h_k}{2}\, \boldsymbol{v}_\phi(\bar{\boldsymbol{x}}_{m_k}, m_k), \tag{33}$$

where $\bar{\boldsymbol{x}}_{m_k} = \boldsymbol{x} - \frac{h_k}{2}\, \boldsymbol{v}_\phi(\boldsymbol{x}, t_k)$.

Thus, $\boldsymbol{f}_\phi(\cdot, t_k, t_{k+1})$ is the composition of two Euler steps of size $h_k/2$. Since $\boldsymbol{v}_\phi(\cdot, t)$ is L-Lipschitz, each Euler step is $(1 + h_k L/2)$-Lipschitz, and hence $\boldsymbol{f}_\phi(\cdot, t_k, t_{k+1})$ is $L_{F,k}$-Lipschitz with

$$L_{F,k} := \Big(1 + \frac{h_k L}{2}\Big)^2. \tag{34}$$

Then,

$$\begin{aligned}
\|\boldsymbol{x}_{t_{k+1}}^{\theta} - \bar{\boldsymbol{x}}_{t_{k+1}}\|_2 &= \|\boldsymbol{f}_\theta(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1}) - \boldsymbol{f}_\phi(\bar{\boldsymbol{x}}_{t_k}, t_k, t_{k+1})\|_2 \\
&\le \|\boldsymbol{f}_\theta(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1}) - \boldsymbol{f}_\phi(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1})\|_2 \\
&\quad + \|\boldsymbol{f}_\phi(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1}) - \boldsymbol{f}_\phi(\bar{\boldsymbol{x}}_{t_k}, t_k, t_{k+1})\|_2 \\
&\le h_k \|\boldsymbol{u}_\theta(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1}) - \boldsymbol{u}_\phi(\boldsymbol{x}_{t_k}^{\theta}, t_k, t_{k+1})\|_2 + L_{F,k} \|\boldsymbol{x}_{t_k}^{\theta} - \bar{\boldsymbol{x}}_{t_k}\|_2.
\end{aligned} \tag{35}$$

Denote $E_k := \mathbb{E}\big[\|\boldsymbol{x}_{t_k}^{\theta} - \bar{\boldsymbol{x}}_{t_k}\|_2^2\big]$ and D_k as in Theorem 1. Then we obtain

$$E_{k+1} \le 2h_k^2 D_k + 2L_{F,k}^2 E_k. \tag{36}$$

Since the two trajectories start from the same noise sample, $E_0 = 0$. Unrolling Eq. (36) gives

$$E_K \le \sum_{k=0}^{K-1} a_k D_k, \qquad a_k := 2h_k^2 \prod_{j=k+1}^{K-1} 2L_{F,j}^2. \tag{37}$$

By the assumption of κ_{roll} and Proposition 2,

$$E_K \le \kappa_{\mathrm{roll}} \Big(\sum_{k=0}^{K-1} a_k \Big) D_{q_\theta}(\boldsymbol{u}_\theta, \boldsymbol{u}_\phi) \le \kappa_{\mathrm{roll}} \Big(\sum_{k=0}^{K-1} a_k \Big) C_\lambda\, \mathcal{L}_{\text{FastOPD}}. \tag{38}$$

The pair $(\boldsymbol{x}_{t_K}^{\theta}, \bar{\boldsymbol{x}}_{t_K})$ constructed from the same initial noise is a valid coupling of p_θ and p_ϕ. Therefore,

$$W_2^2(p_\theta, p_\phi) \le \mathbb{E}\big[\|\boldsymbol{x}_{t_K}^{\theta} - \bar{\boldsymbol{x}}_{t_K}\|_2^2\big] = E_K \le C^2\, \mathcal{L}_{\text{FastOPD}}, \qquad C^2 := \kappa_{\mathrm{roll}}\, C_\lambda \sum_{k=0}^{K-1} a_k. \tag{39}$$

Taking square roots on both sides yields

$$W_2(p_\theta, p_\phi) \le C \sqrt{\mathcal{L}_{\text{FastOPD}}}, \tag{40}$$

hence $\mathcal{L}_{\text{FastOPD}} \to 0$ implies $W_2(p_\theta, p_\phi) \to 0$. □

B EXPERIMENTAL DETAILS

B.1 IMPLEMENTATION DETAILS.

All experiments are implemented within the LeRobot framework (Cadene et al., 2024). We report the public checkpoints used for teacher and student initialization in Tab. 7. For two-time projection, the source and target timesteps are embedded using fixed sinusoidal positional encodings, and the remaining weights of the time projection layer $\psi(\cdot)$ are randomly initialized.

Table 7: Model checkpoints used in the experiments.

Benchmark	Model	Checkpoint
LIBERO	$\pi_{0.5}$	`lerobot/pi05_libero_finetuned_v044`
	SmolVLA	`HuggingFaceVLA/smolvla_libero`
	SmolVLA (OPD Init)	`lerobot/smolvla_base`
RoboTwin 2.0	LingBot-VLA	`robbyant/lingbot-vla-4b-posttrain-robotwin`
	Fast-WAM	`lerobot/fastwam_robotwin_uncond_3cam_384`
	SmolVLA	`lerobot/smolvla_robotwin`
Real-world	MolmoAct2	`allenai/MolmoAct2-BimanualYAM`
	SmolVLA	`lerobot/smolvla_base`

B.2 CONFIGURATION DETAILS.

In Tab. 8, we report training and evaluation configurations for full reproducibility.

Table 8: Training and evaluation configurations for FastOPD.

	LIBERO	RoboTwin 2.0		Real-world
Model				
Teacher (Params)	$\pi_{0.5}$ (3.3B)	LingBot-VLA (4.2B)	Fast-WAM (6B)	MolmoAct2 (5B)
Student (Params)	SmolVLA + ψ (451M)	SmolVLA + ψ (451M)	SmolVLA + ψ (451M)	SmolVLA + ψ (451M)
Optimization				
Optimizer	AdamW	AdamW	AdamW	AdamW
Adam (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
Adam ϵ	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Adam weight decay	10^{-10}	10^{-10}	10^{-10}	10^{-10}
Gradient clipping	10.0	10.0	10.0	10.0
Scheduler	Cosine	Cosine	Cosine	Cosine
Learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	2×10^{-5}
FastOPD training				
Training iterations	10K	50K	50K	20K
Batch size	64	128	128	64
λ	0.1	0.1	0.1	1.0
Time sampling distribution p	Beta(1.5,1)	Beta(1.5,1)	Shifted uniform $(s = 5)$[2]	Beta(1.5,1)
Evaluation				
Time discretization	Uniform	Uniform	Uniform	Uniform
Action execution steps K	10	50	24	30

B.3 COMPUTATIONAL RESOURCES

For full reproducibility, we report the computational resources used in our experiments in Tab. 9. Latency is measured on a single RTX 3090 (App. B), and real-world policies are deployed on a single RTX 4090 (App. B.5).

[2]We sample $u \sim \mathcal{U}(0, 1)$ and apply the flow-shift transformation $t = \frac{su}{1+(s-1)u}$ with $s = 5$, following Fast-WAM (Yuan et al., 2026) implementation.

Table 9: Hardware used for training.

Experiment	Hardware
LIBERO (main)	1× NVIDIA RTX 4090 (24 GB)
LIBERO (ablations)	1× NVIDIA RTX 4090 or RTX 3090 (24 GB each)
RoboTwin 2.0 (LingBot-VLA)	2× NVIDIA RTX 4090 (24 GB each)
RoboTwin 2.0 (Fast-WAM)	1× NVIDIA H200 (141 GB)
Real-world	1× NVIDIA B200 (180 GB)

B.4 Simulation Experimental Details.

Evaluation uses 50 episodes per task (2,000 episodes on LIBERO; 2,500 per setting on RoboTwin 2.0), and all results are from a single training run (seed 100000). Following the latency details of Luan et al. (2026), we measure end-to-end inference latency on a single NVIDIA RTX 3090 (CUDA 12.8, PyTorch 2.11) in bfloat16 with batch size 1, excluding pre- and post-processing, and report the median of 100 runs after 10 warm-up runs.[3]

B.5 Real-world Experimental Details

Robot setup. We follow the real-robot setup of MolmoAct2 (Fang et al., 2026a). Specifically, we use the right arm of a bimanual YAM robot with six joints and a parallel-jaw gripper, controlled by absolute joint-position commands (7-dimensional action space), with the joint velocity capped at 0.75 rad/s (Fig. 8). The policy observes two RGB views (top and right wrist; 480×640, 30 fps; current frame only) together with the 7-dimensional proprioceptive state (six joints and gripper), and receives the language instruction. Actions and states are normalized with quantile normalization for all models (the gripper is not normalized for MolmoAct2). All policies are run on a single NVIDIA RTX 4090, with SmolVLA-based models in float32.

Dataset. We collect teleoperated demonstrations of multiple tasks on the YAM arm and train all policies on the same dataset (Tab. 10). The evaluation is performed on `pnp-plate`. The teacher MolmoAct2 is fine-tuned on this dataset for 14,000 iterations. The student is initialized from SmolVLA, warmed up with flow-matching-based behavior cloning for 10,000 iterations, and then trained with FastOPD for 20,000 iterations. At inference, the type of target objects including a ball and a plate are fixed and the language instruction is fixed as "Pick up the ball and place it on the plate".

Table 10: Real-world dataset statistics. Average episode length is computed at 30 fps.

Task	Language instruction	Episodes	Frames	Avg. Length (s)
`pnp-plate`	"Pick up the ball and place it on the plate."	106	47,002	14.8
`pnp-box`	"Pick up the basketball and place it in the box."	108	47,616	14.7
`open-place`	"Open the box, put the ball inside, and close it."	53	44,780	28.2
	"Open the box, put the basketball inside, and close it."	55	46,834	28.4
	"Open the box, take the ball out, and place it in the pot."	52	41,637	26.7
	"Open the box, take the ball out, and place it on the tray."	50	40,676	27.1
Overall		424	268,545	21.1

Policy execution. Tab. 11 summarizes the execution settings. All policies predict 30-step action chunks (1.0 s) and send commands at 30 Hz. The SmolVLA (warm-up) and our policy share the same synchronous execution scheme. MolmoAct2 is run with its default asynchronous execution, replanning every 0.5 s, skipping the actions already covered by the inference delay, and linearly blending consecutive chunks over 2–10 steps. No real-time chunking or gripper post-processing is applied to any model.

[3] Fast-WAM does not fit in 24 GB memory with its text encoder, so we time the encoder separately and add it.

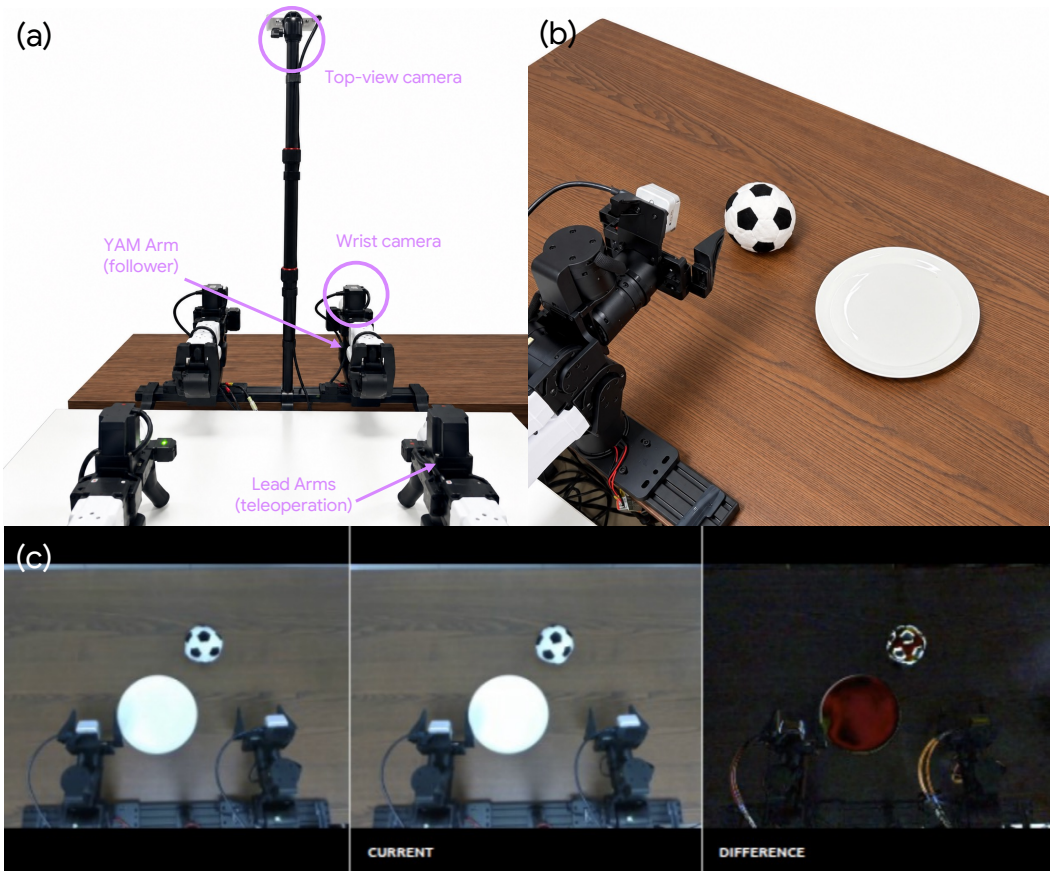


Figure 8: **Robot setting.** (a) Teleoperation setup for data collection, with lead arms, the follower YAM arm, and top-view and wrist cameras. (b) Close-up view of the evaluation setup. (c) Object-position initialization: the current top-camera frame is compared against a reference frame, and their pixel-wise difference is used to place objects at the same initial positions across trials.

Table 11: Real-world policy execution settings. All policies predict 30-step action chunks (1.0 s) and send commands at 30 Hz. For SmolVLA, values are reported for 4 and 10 denoising steps. Inference latency is measured on the robot side, including HTTP communication.

	MolmoAct2	**SmolVLA (warm-up)**	**Ours**
Execution	async (re-plan every 0.5 s)	sync	sync
Sampler	Euler, 10 steps	Euler, 4 / 10 steps	flow map, 4 steps
Actions executed per chunk	$\sim$15	30	30
Chunk blending	linear (2–10 steps)	none	none
Inference latency	300–450 ms	118 / 178 ms	120 ms
Re-planning rate	1.8–1.9 Hz	0.89 / 0.86 Hz	0.87 Hz

Evaluation details. Each policy is evaluated over 50 episodes with a time limit of 30 s. An episode is counted as successful once the ball has been placed entirely within the plate and the gripper has opened. We report the success rate and the task completion time, the latter averaged over successful episodes only, following the Wang et al. (2025) evaluation protocol. To ensure a fair comparison, the initial scene of every episode is matched to the corresponding episode of a reference evaluation session by aligning the current top-camera frame with the reference image (Fig. 8(c)), so that all policies start from identical object configurations.

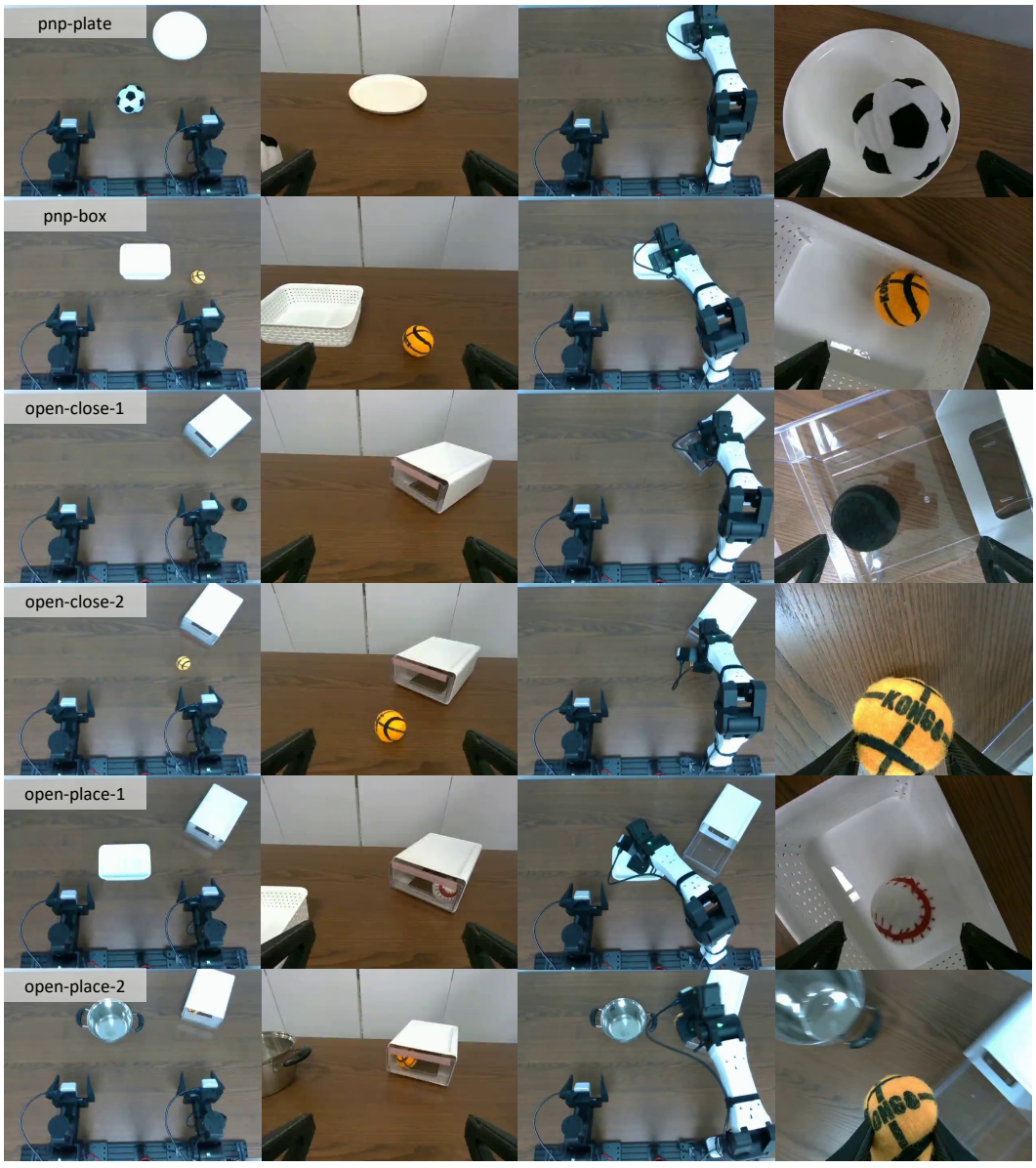


Figure 9: Real-world manipulation task setups on the YAM robot arm. Left two columns show the initial scene and the right show the robot during execution.

C Additional Experiments

Connection with previous works. Although SnapFlow (Luan et al., 2026) adopts a two-step consistency target similar to ours, the two objectives differ in both their anchor and their scope. SnapFlow is a self-distillation method that combines flow matching on ground-truth actions with a shortcut loss defined only on the fixed interval from $t = 1$ to $t = 0$:

$$\mathcal{L}_{\text{SnapFlow}} = \lambda_1 \, \mathcal{L}_{\text{FM}} + (1 - \lambda_1)\lambda_2 \, \mathcal{L}_{\text{shortcut}}, \qquad \mathcal{L}_{\text{shortcut}} = \left\| \boldsymbol{u}_\theta(\boldsymbol{x}_1, 1, 0 \mid c) - \boldsymbol{v}_{\text{tgt}} \right\|_2^2, \tag{41}$$

where $\boldsymbol{v}_{\text{tgt}} = \frac{1}{2}\big[\text{sg}[\boldsymbol{v}_\theta(\boldsymbol{x}_1, 1 \mid c)] + \text{sg}[\boldsymbol{v}_\theta(\hat{\boldsymbol{x}}_{0.5}, 0.5 \mid c)]\big]$ and $\hat{\boldsymbol{x}}_{0.5} = \boldsymbol{x}_1 - 0.5\,\text{sg}[\boldsymbol{v}_\theta(\boldsymbol{x}_1, 1 \mid c)]$. In contrast, FastOPD enforces self-consistency over arbitrary intervals (s, t), yielding a flow map that supports any number of sampling steps, and anchors the target to the teacher through OPFD rather than to the data through flow matching. SnapFlow can thus be viewed as a teacher-free counterpart

of our objective restricted to a single interval, which allows us to isolate the contribution of teacher supervision.

Student initialization. In Sec. 4.3, we studied on-policy distillation (OPD) as an alternative initialization strategy for the flow map. Specifically, we first train the SmolVLA base model with OPD for 5,000 iterations (Stage 1), and then fine-tune it into a flow map with the FastOPD objective for 10,000 iterations (Stage 2). Tab. 12 reports the per-suite results, and Fig. 10 shows the success rate over training iterations. As in Fig. 4, FastOPD consistently improves performance over the course of training under this OPD-based initialization as well.

Table 12: Per-suite results of OPD-based initialization. Success rates (%) on LIBERO.

Method	NFE	Spatial	Object	Goal	Long	Avg.
OPD init.	1	78.0	94.2	78.2	40.2	72.7
	2	84.4	95.4	87.8	45.0	78.2
	4	58.0	59.8	53.8	5.0	44.2
	10	82.6	94.6	89.0	48.2	78.6
+ FastOPD	1	84.2	92.8	90.8	54.0	80.5
	2	84.6	95.2	94.2	57.8	**83.0**
	4	83.6	97.2	91.4	56.4	<u>82.2</u>
	10	80.4	92.8	93.4	60.2	81.7

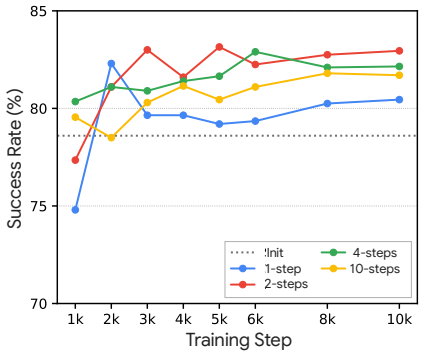


Figure 10: Success rate (%) on LIBERO over training iterations with OPD-based initialization.

D Few-step Distillation Algorithms

In this section, we provide detailed descriptions of the few-step distillation algorithms leveraged in our experiments as baselines.

D.0.1 Distribution Matching Distillation

Referring to OneDP (Wang et al., 2025), we design DMD as a one-step generator, and jointly train with a fake-score critic $\boldsymbol{v}_\psi(\boldsymbol{x}, t)$ of the same architecture, initialized from the same checkpoint as $\boldsymbol{v}_\theta$. Here, generator and critic are updated alternately at every step.

Generator. A sample is produced with a single Euler step of the student's own PF-ODE from pure noise $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$,

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_1 - \boldsymbol{v}_\theta(\boldsymbol{x}_1, 1). \tag{42}$$

From the re-noised sample $\boldsymbol{x}_t$ from $\hat{\boldsymbol{x}}_0$, the teacher and critic denoised estimates are formed as

$$\hat{\boldsymbol{x}}_0^{\text{real}} = \boldsymbol{x}_t - t\,\boldsymbol{v}_\phi(\boldsymbol{x}_t, t), \qquad \hat{\boldsymbol{x}}_0^{\text{fake}} = \boldsymbol{x}_t - t\,\boldsymbol{v}_\psi(\boldsymbol{x}_t, t). \tag{43}$$

The distribution-matching direction is then defined as $\mathbf{g}_0 = \hat{\boldsymbol{x}}_0^{\text{fake}} - \hat{\boldsymbol{x}}_0^{\text{real}}$. Here, we normalize it per sample by the mean absolute magnitude of the teacher's own correction,

$$\mathbf{g} = \frac{\mathbf{g}_0}{\text{mean}\big(\big|\hat{\boldsymbol{x}}_0^{\text{real}} - \text{sg}[\hat{\boldsymbol{x}}_0]\big|\big) + \epsilon}, \tag{44}$$

where $\text{mean}(\cdot)$ averages over the action-chunk and action-dimension axes, for training stability. The generator is trained with the surrogate loss whose gradient with respect to $\hat{\boldsymbol{x}}_0$ equals $\mathbf{g}$:

$$\mathcal{L}_{\text{gen}}(\theta) = \tfrac{1}{2}\big\|\hat{\boldsymbol{x}}_0 - \text{sg}[\hat{\boldsymbol{x}}_0 - \mathbf{g}]\big\|_2^2. \tag{45}$$

Critic. The fake-score network $\boldsymbol{v}_\psi$ is trained with ordinary denoising score matching on the generator's output:

$$\mathcal{L}_{\text{crit}}(\psi) = \mathbb{E}_{t',\boldsymbol{\epsilon}}\Big[\big\|\boldsymbol{v}_\psi\big(\boldsymbol{x}_{t'}, t'\big) - \big(\boldsymbol{\epsilon} - \text{sg}[\hat{\boldsymbol{x}}_0]\big)\big\|_2^2\Big]. \tag{46}$$

where $\boldsymbol{x}_{t'} = (1-t')\,\text{sg}[\hat{\boldsymbol{x}}_0] + t'\boldsymbol{\epsilon}$.

Algorithm 2 Training Algorithm for DMD

Require: Dataset $\mathcal{D}$; teacher $\boldsymbol{v}_\phi$; generator $\boldsymbol{v}_\theta$; critic $\boldsymbol{v}_\psi$

for each training iteration **do**
- $(\boldsymbol{x}_0, c) \sim \mathcal{D}$, $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\hat{\boldsymbol{x}}_0 \leftarrow \boldsymbol{x}_1 - \boldsymbol{v}_\theta(\boldsymbol{x}_1, 1 \mid c)$ ▷ *one-step generation*
- $t \sim p(t)$, $\boldsymbol{x}_1' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; $\quad \boldsymbol{x}_t \leftarrow (1-t)\,\text{sg}[\hat{\boldsymbol{x}}_0] + t\,\boldsymbol{x}_1'$
- $\hat{\boldsymbol{x}}_0^{\text{real}} \leftarrow \boldsymbol{x}_t - t\,\boldsymbol{v}_\phi(\boldsymbol{x}_t, t \mid c)$, $\quad \hat{\boldsymbol{x}}_0^{\text{fake}} \leftarrow \boldsymbol{x}_t - t\,\boldsymbol{v}_\psi(\boldsymbol{x}_t, t \mid c)$
- $\mathbf{g} \leftarrow \big(\hat{\boldsymbol{x}}_0^{\text{fake}} - \hat{\boldsymbol{x}}_0^{\text{real}}\big) \big/ \big(\text{mean}(|\hat{\boldsymbol{x}}_0^{\text{real}} - \text{sg}[\hat{\boldsymbol{x}}_0]|) + \epsilon\big)$
- $\mathcal{L}_{\text{gen}} \leftarrow \frac{1}{2}\|\hat{\boldsymbol{x}}_0 - \text{sg}[\hat{\boldsymbol{x}}_0 - \mathbf{g}]\|_2^2$
- $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{gen}}$ ▷ *update generator*
- $t' \sim p(t)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\boldsymbol{x}_{t'} \leftarrow (1-t')\,\text{sg}[\hat{\boldsymbol{x}}_0] + t'\boldsymbol{\epsilon}$
- $\mathcal{L}_{\text{crit}} \leftarrow \|\boldsymbol{v}_\psi(\boldsymbol{x}_{t'}, t' \mid c) - (\boldsymbol{\epsilon} - \text{sg}[\hat{\boldsymbol{x}}_0])\|_2^2$
- $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_{\text{crit}}$ ▷ *update critic*

end for

D.0.2 Consistency Trajectory Model

We follow the Consistency Trajectory Model (CTM) formulation of (Kim et al., 2024), which generalizes the original Consistency Distillation (Song et al., 2023) by allowing anytime-to-anytime jump with a denoising score-matching (DSM) regularizer. We train the two-time flow map $\boldsymbol{f}_\theta(\boldsymbol{x}, s, t) = \boldsymbol{x} + (t-s)\boldsymbol{u}_\theta(\boldsymbol{x}, s, t)$ together with an EMA target network θ^-, updated as $\theta^- \leftarrow \mu\theta^- + (1-\mu)\theta$.

For sampled timesteps $1 \geq s \geq r \geq 0$ and an intermediate solver time $u \sim \text{Unif}[r, s]$, with $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{x}_0 \sim p_0$, and $\boldsymbol{x}_s = (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$, the student first jumps from s to r,

$$\hat{\boldsymbol{x}}_r^\theta = \boldsymbol{f}_\theta(\boldsymbol{x}_s, s, r), \tag{47}$$

while the target integrates the teacher's PF-ODE from s to u with a 1-step Heun solver, $\boldsymbol{x}_u = \text{ODE}_\phi(\boldsymbol{x}_s; s, u)$, and then estimates from time u to r with the EMA network,

$$\boldsymbol{x}_r^- = \boldsymbol{f}_{\theta^-}(\boldsymbol{x}_u, u, r). \tag{48}$$

Both are then projected to the clean timestep $t = 0$ through a stop-gradient student $G_{\text{sg}}(\cdot, r, 0) := \boldsymbol{f}_{\text{sg}[\theta]}(\cdot, r, 0)$, giving $\hat{\boldsymbol{x}}_0^\theta = G_{\text{sg}}(\hat{\boldsymbol{x}}_r^\theta, r, 0)$ and $\boldsymbol{x}_0^- = G_{\text{sg}}(\boldsymbol{x}_r^-, r, 0)$. Thus, the CTM loss is defined as

$$\mathcal{L}_{\text{CTM}}(\theta) = \mathbb{E}\big\|\hat{\boldsymbol{x}}_0^\theta - \text{sg}[\boldsymbol{x}_0^-]\big\|_2^2, \tag{49}$$

and we add a DSM regularizer anchoring the student's instantaneous velocity $\boldsymbol{v}_\theta(\boldsymbol{x}_s, s)$ to the teacher's,[4]

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}\big\|\boldsymbol{v}_\theta(\boldsymbol{x}_s, s) - \text{sg}[\boldsymbol{v}_\phi(\boldsymbol{x}_s, s)]\big\|_2^2, \tag{50}$$

so the total loss is $\mathcal{L}(\theta) = \mathcal{L}_{\text{CTM}}(\theta) + w_{\text{dsm}}\,\mathcal{L}_{\text{DSM}}(\theta)$.

[4]The regularizer can instead target the ground-truth flow velocity $\boldsymbol{x}_1 - \boldsymbol{x}_0$ (dataset-supervised DSM, as in (Kim et al., 2024)); we use the teacher-supervised variant.

Algorithm 3 Training Algorithm for CTM

Require: Dataset $\mathcal{D}$; teacher $\boldsymbol{v}_\phi$; student flow map $\boldsymbol{u}_\theta$; EMA target θ^-; decay μ; DSM weight w_{dsm}

for each training iteration **do**
  $(\boldsymbol{x}_0, c) \sim \mathcal{D}$, $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, sample $1 \geq s \geq r \geq 0$, $u \sim \mathrm{Unif}[r, s]$
  $\boldsymbol{x}_s \leftarrow (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$
  $\hat{\boldsymbol{x}}_r^\theta \leftarrow \boldsymbol{f}_\theta(\boldsymbol{x}_s, s, r \mid c)$
  $\boldsymbol{x}_u \leftarrow \mathrm{ODE}_\phi(\boldsymbol{x}_s; s, u \mid c)$
  $\boldsymbol{x}_r^- \leftarrow \boldsymbol{f}_{\theta^-}(\boldsymbol{x}_u, u, r \mid c)$
  $\hat{\boldsymbol{x}}_0^\theta \leftarrow \boldsymbol{f}_{\mathrm{sg}[\theta]}(\hat{\boldsymbol{x}}_r^\theta, r, 0 \mid c)$; $\boldsymbol{x}_0^- \leftarrow \boldsymbol{f}_{\mathrm{sg}[\theta]}(\boldsymbol{x}_r^-, r, 0 \mid c)$
  $\mathcal{L}_{\mathrm{CTM}} \leftarrow \|\hat{\boldsymbol{x}}_0^\theta - \mathrm{sg}[\boldsymbol{x}_0^-]\|_2^2$
  $\mathcal{L}_{\mathrm{DSM}} \leftarrow \|\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, s \mid c) - \mathrm{sg}[\boldsymbol{v}_\phi(\boldsymbol{x}_s, s \mid c)]\|_2^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\big(\mathcal{L}_{\mathrm{CTM}} + w_{\mathrm{dsm}}\mathcal{L}_{\mathrm{DSM}}\big)$
  $\theta^- \leftarrow \mu\theta^- + (1-\mu)\theta$
end for

D.0.3 IMPROVED MEANFLOW

MeanFlow (Geng et al., 2025) parameterizes the average velocity $\boldsymbol{u}(\boldsymbol{x}, s, t) := \frac{1}{s-t}\int_t^s \boldsymbol{v}(\boldsymbol{x}_\tau, \tau)\, d\tau$ and train it through the identity obtained by differentiating the definition along s,

$$\boldsymbol{u}(\boldsymbol{x}_s, s, t) = \boldsymbol{v}(\boldsymbol{x}_s, s) - (s-t)\,\frac{d}{ds}\boldsymbol{u}(\boldsymbol{x}_s, s, t), \qquad \frac{d}{ds}\boldsymbol{u} = \partial_{\boldsymbol{x}}\boldsymbol{u}\cdot\boldsymbol{v}(\boldsymbol{x}_s, s) + \partial_s\boldsymbol{u}. \tag{51}$$

In the original formulation, $\boldsymbol{v}(\boldsymbol{x}_s, s)$ in Eq. (51) is taken to be the student's own instantaneous velocity $\boldsymbol{v}_\theta(\boldsymbol{x}_s, s) := \boldsymbol{u}_\theta(\boldsymbol{x}_s, s, s)$, so the regression target is itself computed by the network being optimized. Improved MeanFlow (iMF) (Geng et al., 2026) addresses this issue by reformulating the identity to remove this self-reference.

We follow this reformulation and further adapt it to distillation by replacing the self-referential term with the frozen teacher velocity, $\boldsymbol{v}_\phi(\boldsymbol{x}_s, s)$, used both as the target and the tangent direction of the total derivative.

Given $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{x}_0 \sim p_0$, and interpolant $\boldsymbol{x}_s$, the total derivative is computed by a Jacobian–Vector Product (JVP) of $\boldsymbol{u}_\theta$ at $(\boldsymbol{x}_s, s, t)$ along the tangent $(\boldsymbol{v}_\phi, 1, 0)$:

$$\dot{\boldsymbol{u}}_\theta := \mathrm{JVP}\big(\boldsymbol{u}_\theta;\ (\boldsymbol{x}_s, s, t),\ (\boldsymbol{v}_\phi, 1, 0)\big), \tag{52}$$

The model then predicts one-step velocity as follow:

$$\hat{\boldsymbol{v}}_\theta = \boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t) + (s-t)\,\mathrm{sg}[\dot{\boldsymbol{u}}_\theta]. \tag{53}$$

Note that MF loss reduces to standard flow matching when $t = s$. The loss regresses $\hat{\boldsymbol{v}}_\theta$ onto the teacher target with an elementwise Pseudo-Huber loss,

$$\mathcal{L}_{\mathrm{iMF}}(\theta) = \mathbb{E}\big[\rho_\delta(\hat{\boldsymbol{v}}_\theta - \boldsymbol{v}_\phi)\big], \qquad \rho_\delta(e) = \delta^2\Big(\sqrt{1 + e^2/\delta^2} - 1\Big), \tag{54}$$

applied element-wise and averaged over the action dimensions.

Algorithm 4 Training Algorithm for iMF

Require: Dataset $\mathcal{D}$; teacher $\boldsymbol{v}_\phi$; student flow map $\boldsymbol{u}_\theta$; FM prob. $p_{t=s}$; Huber scale δ

for each training iteration **do**
  $(\boldsymbol{x}_0, c) \sim \mathcal{D}$, $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, sample $1 \geq s \geq t \geq 0$
  With prob. $p_{t=s}$: $t \leftarrow s$
  $\boldsymbol{x}_s \leftarrow (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1$
  $\boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t \mid c)$, $\dot{\boldsymbol{u}}_\theta \leftarrow \mathrm{sg}\left[\mathrm{JVP}\big(\boldsymbol{u}_\theta(\cdot,\cdot,t \mid c);\ (\boldsymbol{x}_s, s),\ (\boldsymbol{v}_\phi, 1)\big)\right]$
  $\hat{\boldsymbol{v}}_\theta \leftarrow \boldsymbol{u}_\theta(\boldsymbol{x}_s, s, t \mid c) + (s-t)\,\dot{\boldsymbol{u}}_\theta$
  $\mathcal{L}_{\mathrm{iMF}} \leftarrow \rho_\delta(\hat{\boldsymbol{v}}_\theta - \boldsymbol{v}_\phi)$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_{\mathrm{iMF}}$
end for

E Detailed Results in Simulation Environments

Table 13: Per-task success rates (%) on RoboTwin 2.0 at 10 denoising steps, under clean and randomized settings. Ours (X) denotes our method trained with X as the teacher.

Task	LingBot-VLA		Fast-WAM		SmolVLA		Ours (LingBot-VLA)		Ours (Fast-WAM)	
	Clean	Random	Clean	Random	Clean	Random	Clean	Random	Clean	Random
adjust_bottle	98	90	100	100	100	98	100	98	100	98
beat_block_hammer	78	78	100	96	66	66	56	40	76	78
blocks_ranking_rgb	96	92	100	100	50	46	50	58	30	30
blocks_ranking_size	74	66	88	96	20	22	28	26	20	10
click_alarmclock	70	42	100	100	90	94	34	50	92	96
click_bell	62	78	100	100	96	92	62	54	96	86
dump_bin_bigbin	92	98	100	96	96	70	92	84	90	82
grab_roller	100	98	100	100	96	96	100	100	100	100
handover_block	88	82	96	90	28	22	18	32	28	28
handover_mic	100	88	100	98	80	74	76	82	84	88
hanging_mug	48	50	64	56	10	16	4	12	6	8
lift_pot	100	100	100	100	84	72	62	68	80	80
move_can_pot	66	86	88	98	22	44	18	44	10	32
move_pillbottle_pad	98	94	96	100	54	54	50	56	52	56
move_playingcard_away	100	100	98	98	78	84	88	88	68	80
move_stapler_pad	56	66	74	66	24	14	30	22	26	18
open_laptop	96	94	94	100	88	86	78	78	72	88
open_microwave	58	38	48	46	18	36	20	54	22	30
pick_diverse_bottles	82	76	82	90	42	32	60	44	50	40
pick_dual_bottles	94	88	100	100	50	42	50	42	64	64
place_a2b_left	84	80	100	98	38	48	46	42	40	42
place_a2b_right	76	82	94	98	40	36	38	46	38	44
place_bread_basket	84	90	94	90	68	68	66	56	60	60
place_bread_skillet	92	70	92	92	62	54	66	62	68	70
place_burger_fries	94	94	94	100	96	84	96	92	98	90
place_can_basket	82	72	76	68	62	44	30	24	32	8
place_cans_plasticbox	100	100	100	100	82	84	58	52	88	82
place_container_plate	98	96	100	96	90	94	98	96	90	88
place_dual_shoes	80	84	94	88	50	62	58	76	42	50
place_empty_cup	100	98	100	100	96	98	98	98	86	84
place_fan	88	80	98	96	38	50	38	44	42	56
place_mouse_pad	86	90	82	84	26	42	34	42	30	36
place_object_basket	92	84	88	82	68	62	82	80	70	48
place_object_scale	82	82	100	94	46	38	50	58	36	52
place_object_stand	92	96	98	98	60	82	62	82	70	86
place_phone_stand	80	96	92	96	48	74	46	76	60	70
place_shoe	96	96	88	92	86	88	90	88	76	78
press_stapler	94	96	90	98	78	74	66	64	64	72
put_bottles_dustbin	90	84	90	92	36	42	34	44	30	38
put_object_cabinet	90	78	88	90	60	60	60	54	72	64
rotate_qrcode	88	88	94	82	64	64	64	68	56	52
scan_object	96	94	90	94	60	70	46	48	54	54
shake_bottle	100	100	100	100	100	98	98	98	100	96
shake_bottle_horizontally	100	98	100	100	98	98	98	98	96	98
stack_blocks_three	92	94	94	100	4	20	4	6	8	16
stack_blocks_two	100	100	100	100	40	50	64	66	42	58
stack_bowls_three	86	72	82	76	48	56	36	46	52	50
stack_bowls_two	96	98	92	98	70	84	72	74	80	74
stamp_seal	88	92	90	92	26	34	40	30	18	32
turn_switch	56	56	68	68	40	48	32	36	50	38
Average	86.76	84.88	91.92	91.84	59.44	61.32	56.92	59.56	58.28	59.56

Table 14: Per-task success rates (%) of our method with LingBot-VLA and Fast-WAM as teachers on RoboTwin 2.0 at 1, 2, and 4 denoising steps, under clean and randomized settings. Ours (X) denotes our method trained with X as the teacher.

Task	Ours (LingBot-VLA)						Ours (Fast-WAM)					
	1-step		2-step		4-step		1-step		2-step		4-step	
	Clean	Random	Clean	Random	Clean	Random	Clean	Random	Clean	Random	Clean	Random
adjust_bottle	94	96	100	98	100	98	94	98	100	96	100	98
beat_block_hammer	72	58	84	68	66	66	56	50	76	68	80	74
blocks_ranking_rgb	34	32	38	44	52	56	34	32	44	32	32	32
blocks_ranking_size	6	6	20	18	28	16	12	8	12	10	20	14
click_alarmclock	24	46	22	38	36	54	90	82	82	92	90	90
click_bell	94	92	82	88	66	74	100	100	98	94	98	92
dump_bin_bigbin	60	56	76	72	84	78	76	76	70	68	76	86
grab_roller	96	98	96	98	98	96	100	98	100	100	100	98
handover_block	12	12	14	16	26	22	20	12	22	24	30	20
handover_mic	58	66	70	64	72	70	58	68	72	74	86	92
hanging_mug	6	6	10	12	6	18	2	4	4	2	6	10
lift_pot	78	80	92	86	84	86	70	74	74	74	82	76
move_can_pot	38	32	22	46	32	46	22	26	26	40	22	20
move_pillbottle_pad	50	40	70	70	72	68	44	52	60	62	60	56
move_playingcard_away	70	82	86	84	86	86	68	78	72	78	76	84
move_stapler_pad	20	22	20	20	24	20	14	16	16	22	16	32
open_laptop	92	94	88	90	82	84	78	86	76	90	76	92
open_microwave	6	12	10	12	12	26	10	4	14	20	14	26
pick_diverse_bottles	56	36	66	44	58	44	48	40	48	36	50	42
pick_dual_bottles	74	58	60	62	58	52	52	64	70	58	68	68
place_a2b_left	38	42	30	44	38	44	32	40	34	42	40	42
place_a2b_right	28	34	36	38	38	38	28	38	46	44	40	50
place_bread_basket	52	52	58	56	64	60	64	56	70	64	70	64
place_bread_skillet	80	64	70	58	76	68	70	52	76	62	68	68
place_burger_fries	96	90	92	92	98	94	90	90	98	96	98	96
place_can_basket	40	38	58	40	44	36	42	24	34	22	26	16
place_cans_plasticbox	58	60	82	78	80	76	52	44	72	78	84	78
place_container_plate	84	90	96	88	90	94	88	92	76	82	92	92
place_dual_shoes	52	52	48	70	56	68	50	54	56	62	58	56
place_empty_cup	78	84	94	96	90	98	72	56	70	68	86	76
place_fan	16	16	40	48	48	44	24	14	38	52	44	44
place_mouse_pad	28	24	26	46	28	50	20	28	30	36	38	26
place_object_basket	66	58	80	72	70	74	44	30	56	32	66	56
place_object_scale	40	46	52	60	54	56	32	34	38	58	44	50
place_object_stand	66	80	66	82	70	86	70	64	70	88	74	90
place_phone_stand	26	52	44	58	54	72	40	48	40	58	48	62
place_shoe	82	84	84	86	84	90	66	80	74	82	68	80
press_stapler	86	78	74	74	72	74	80	84	74	72	74	70
put_bottles_dustbin	8	10	20	10	18	34	6	4	14	22	34	38
put_object_cabinet	40	28	52	46	56	54	52	46	68	62	58	52
rotate_qrcode	66	54	60	64	60	72	58	52	64	54	52	58
scan_object	58	48	58	64	52	64	64	50	54	54	50	62
shake_bottle	98	98	100	100	96	96	98	96	96	98	98	96
shake_bottle_horizontally	96	94	98	98	100	98	96	94	100	98	94	94
stack_blocks_three	6	2	2	4	10	14	10	0	8	12	6	10
stack_blocks_two	20	16	32	36	46	56	30	36	42	50	56	58
stack_bowls_three	6	12	36	30	50	34	16	14	28	24	42	40
stack_bowls_two	68	46	74	70	76	86	62	48	58	70	84	68
stamp_seal	12	20	20	38	30	46	14	18	16	36	10	28
turn_switch	40	48	46	38	40	26	38	36	40	40	40	38
Average	51.48	50.88	57.08	58.28	58.60	61.24	51.12	49.80	55.52	57.16	58.48	59.20

F QUALITATIVE RESULTS

We provide qualitative results in simulation (Fig. 11) and on the real-robot (Fig. 12) experiments.

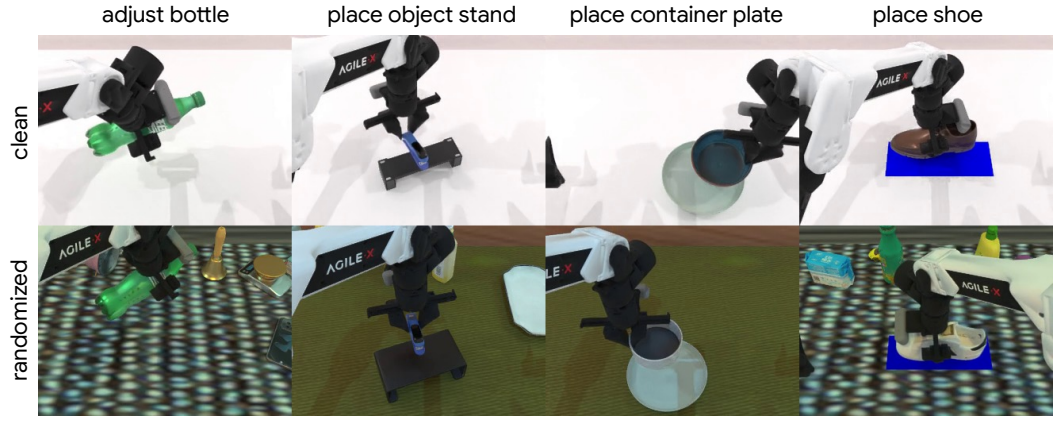


Figure 11: **Qualitative results on RoboTwin 2.0.** FastOPD distilled from LingBot-VLA (NFE=1) executing four tasks under the clean (top) and randomized (bottom) settings, where the randomized setting introduces cluttered distractors, textured tabletops, and varied backgrounds.

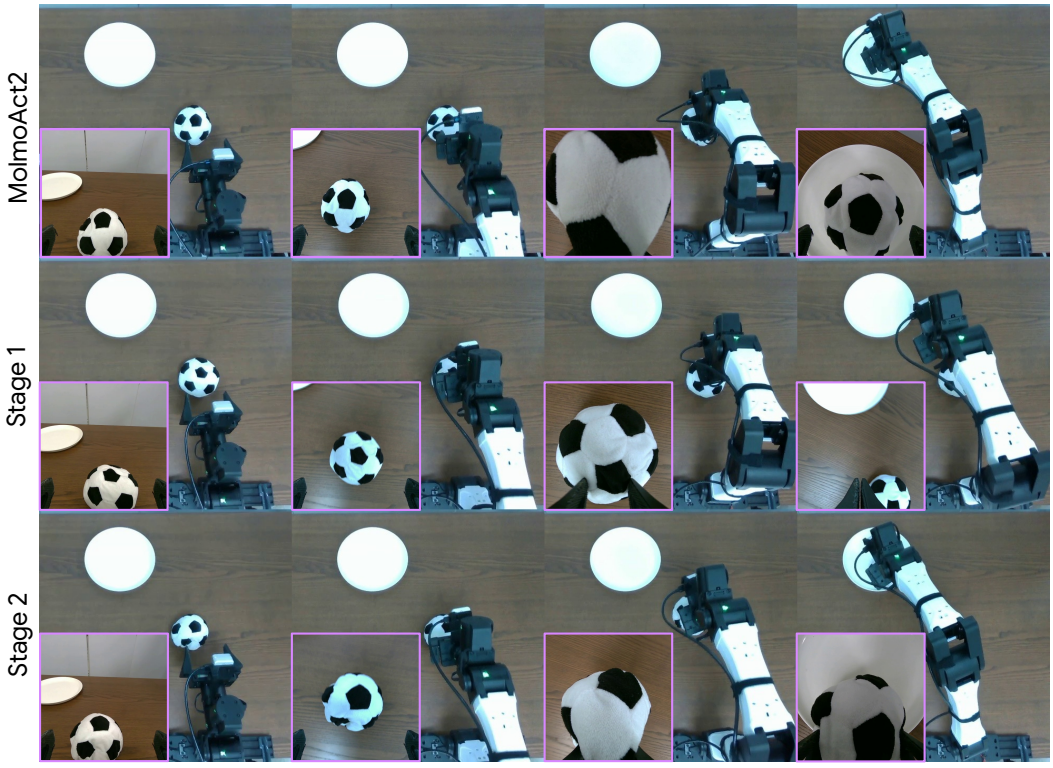


Figure 12: **Qualitative results on the real-world task.** Rollouts of the MolmoAct2 teacher and the student after Stage 1 (initialization) and Stage 2 (FastOPD) on the YAM robot arm, in which the robot picks up a ball and places it on a plate. Time progresses from left to right; insets show the wrist-camera view.