FReD-VLA: A Vision-Language-Action Model with Flow Matching-Initialized Diffusion

Anonymous authors
Paper under double-blind review

Abstract

Deploying vision-language-action (VLA) models for bimanual manipulation in real-world environments requires robots to generate accurate actions with low latency to respond to changing surroundings. State-of-the-art methods usually adopt diffusion- or flow matching-based approaches for continuous action generation. Despite promising results, balancing accuracy and latency still remains challenging. Diffusion models can generate accurate actions, but repeated denoising increases computation and latency. One-step flow matching is efficient, but its actions tend to be inaccurate for fine-grained manipulation. In this paper, we propose a Vision-language-Action model with Flow Matching initialized Diffusion (FReD-VLA), which combines fast flow-based generation with diffusion-based correction. Given visual observations, language instructions, and robot states, a flow matching model first generates a coarse action sequence. The coarse action sequence is then refined by a diffusion model through Action Distribution Calibrator, enabling refinement from a well-initialized intermediate diffusion state rather than the fully-corrupted beginning state. Empirically, on Maniskill PickCube-v1, our model achieves a mean minimum end-effector-to-cube distance of 10.8 mm, with 75.3% reduction than the 43.7 mm measured for pure Diffusion. In offline evaluation, one diffusion step reduces the joint mean absolute error of a one-step flow proposal from 1.176° to 0.620° ($\downarrow$ 47.3%). With one flow evaluation and one diffusion evaluation, FReD-VLA achieves an action-generation latency of 37.0 ms, providing a $4.1\times$ speedup over five-step diffusion with competitive action accuracy. Real-time experiments further demonstrate that the robot can perform dynamic bimanual assembly after adaptation with about 20–30 human demonstrations per behavior.

1 Introduction

Vision-language-action (VLA) models connect visual observations and language instructions to robot actions. They allow a robot to use what it sees and what a user asks to decide how to move (Kim et al., 2025). This provides a useful basis for bimanual manipulation, where two arms must work together to complete the downstream tasks such as holding, aligning, or assembling objects. Practical deployment requires generating accurate actions with low inference latency. For example, a robot which is working on a continuously running production line must insert a connector into a socket on a moving part. Small alignment errors can prevent insertion. Meanwhile if action generation takes too long, the part may move out of reach before the robot can finish assembly.

Diffusion models and flow matching are two widely used approaches for continuous action generation. Diffusion Policy generates action sequences through repeated denoising, conditioned on observations (Chi et al., 2023). RDT-1B extends this approach to language-conditioned bimanual manipulation (Liu et al., 2025). At each denoising step, the model runs its action network to update the noisy action sequence. These steps run in sequence, so more steps increase the time needed to generate actions. Flow matching learns a velocity field that transforms noise into an action sequence (Lipman et al., 2023). VLA models such as π_0 use this approach for continuous action generation (Black et al., 2025). One-step integration reduces computation but can limit action accuracy. In our offline comparison, five-step diffusion produces lower joint errors but has higher latency. One-step flow matching is faster but less accurate. These results motivate combining the speed of flow matching with the accuracy of diffusion.

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.

Recent work has explored how to combine these two generation models for efficient control. Mix Diffusion Policy uses a hierarchical design (Jing et al., 2026). A 3D Diffusion Policy provides task-level guidance, and a flow-based policy generates executable actions from this guidance in one step. The two models therefore operate at different levels of the policy. HybridFlow takes a more direct coarse-to-fine approach (Dong et al., 2026). It further improves inference efficiency through two stages of Flow-based generation. A coarse MeanFlow prediction is re-noised and then corrected by ReFlow, using only two network evaluations. However, the rationale for recovering a suitable refinement state through ReNoise remains heuristic, and the noise strength is chosen empirically. Failures also remain at contact stages that require precise actions, including slipping or tumbling when the gripper touches an object edge. Accurate refinement under low inference budget remains an open challenge.

Following this coarse-to-fine approach, we explore diffusion as a refiner for actions generated by flow matching. Diffusion learns to recover clean actions from noisy inputs. Diffusion policies can also generate precise actions for fine manipulation (Chi et al., 2023; Liu et al., 2025). We therefore propose Flow-initialized Refinement with Diffusion (FReD-VLA) for efficient bimanual action generation. Flow first generates a coarse action sequence from visual observations, language instructions, and robot states. This sequence provides an initial motion estimate, but its joint commands may still contain prediction inaccuracy. The diffusion stage then revises the joint commands across the sequence before execution. It uses the same task conditions and action patterns learned from demonstrations to guide these updates. This correction aims to reduce errors that can affect precise alignment and contact. We refine the coarse action sequence generated by Flow using only one or a few diffusion network evaluations. Here, one network evaluation means running the action network once on its input to obtain a prediction. This short denoising process aims to improve action accuracy without generating actions from pure noise.

Combining these two models in one policy requires a compatible connection between them. Both models can generate action sequences, but the diffusion denoiser has specific input requirements. At a given timestep, it expects an action state with a specific signal scale and noise level. These are defined by the forward Gaussian noise kernel used during training. Flow matching, however, produces a coarse action sequence without applying this forward process. Directly passing this sequence to the denoiser can therefore lead to an input mismatch. To address this mismatch, we introduce an Action Distribution Calibrator (ADC) between the Flow generator and the diffusion refiner. ADC uses the diffusion model's forward noise kernel to construct an intermediate state from the Flow proposal. It scales the proposal and adds Gaussian noise according to the schedule at the selected timestep. The resulting state retains information from the Flow proposal and provides the starting point for diffusion refinement. ADC is computed directly and requires no additional network evaluation. In our fast configuration, the Flow and diffusion networks are each run once. We vary the starting timestep to study how the noise level affects the balance between preserving the proposal and enabling correction.

We evaluate FReD-VLA through ManiSkill simulation, offline action prediction on real-robot data, and dynamic bimanual assembly. On ManiSkill PickCube-v1, our Flow-conditioned variant achieves a mean minimum end-effector-to-cube distance of 10.8 mm. This is 75.3% lower than the 43.7 mm measured for pure diffusion. In the offline comparison, the hybrid model reuses the checkpoints of the standalone Flow and diffusion baselines. One diffusion refinement step reduces the joint mean absolute error of a one-step Flow proposal from $1.176°$ to $0.620°$, a 47.3% reduction. Action generation takes 37.0 ms, excluding shared multimodal encoding. This gives a $4.1\times$ speedup over five-step diffusion at 151.9 ms, with comparable joint MAE ($0.620°$ versus $0.589°$). At nearly the same latency as eight-step Flow, FReD-VLA also reduces joint MAE by 24.5%. A real-robot demonstration further shows dynamic bimanual assembly with moving parts.

Our contributions are as follows:

- We introduce FReD-VLA, which combines a fast Flow proposal with diffusion refinement of the same action sequence. The design reduces repeated diffusion computation while retaining action accuracy.
- We develop an Action Distribution Calibrator that connects the two generators using the diffusion model's forward noising rule. It constructs a refinement input with the signal scaling and noise level specified by the chosen timestep.

- Extensive empirical and offline evaluations demonstrate the effectiveness of the proposed method, delivering a better trade-off between accuracy and latency and improving the practical usability.

2 Related Work

2.1 Continuous action generation for robot manipulation

Action learning in VLA models builds on imitation learning, where robots learn how to act from demonstrations (Kim et al., 2025). Manipulation policies must use visual and other observations to generate actions that remain consistent over time. Bimanual tasks also require the two arms to move together. ACT learns long-horizon bimanual tasks through action chunking and a conditional latent-variable model (Zhao et al., 2023). Generalist VLA models build on this learning setting by using pretrained vision-language representations for control. OpenVLA is an open-source example (Kim et al., 2025). Diffusion Policy generates continuous action sequences through repeated denoising and can represent multiple ways to complete a task (Chi et al., 2023). RDT-1B extends diffusion-based action generation to a pretrained Transformer for bimanual manipulation (Liu et al., 2025).

Building on this multimodal diffusion backbone, we study how to use the action-generation budget during task adaptation. We retain the pretrained diffusion action model and use it to refine a coarse proposal. This lets the diffusion stage focus on local correction, while a separate Flow model provides the initial action sequence.

2.2 Flow Matching and efficient robot policies

Flow Matching learns a velocity field that moves samples from a source distribution toward a target distribution (Lipman et al., 2023). Rectified Flow uses straight paths between source and target samples to support efficient numerical integration (Liu et al., 2023). These methods provide a basis for fast action generation. For example, π_0 uses Flow Matching to generate continuous robot actions (Black et al., 2025). VITA uses visual representations as the source of the flow, removing the need to repeatedly condition generation on those representations (Gao et al., 2026). HybridFlow divides generation into a coarse MeanFlow prediction, ReNoise, and local ReFlow correction (Dong et al., 2026).

These studies motivate giving coarse generation and local correction different roles. FReD-VLA assigns these roles to two separately trained models. A conditional Flow model generates the coarse action sequence. A pretrained diffusion model then refines it. ADC connects the two stages using the diffusion model's own forward noising rule.

2.3 Informed priors and hierarchical generative refinement

Conditional generation does not always need to start from pure Gaussian noise. A starting state that already contains useful information can shorten the generation process. Residual Shifting applies this idea to image restoration. It builds a short stochastic process between low-quality and high-quality images and also admits a conditional-flow interpretation (Yue et al., 2024). Generator Matching provides a common framework for diffusion and Flow Matching based on Markov processes (Holderrieth et al., 2025). This framework describes how both methods move probability distributions. In robotics, ResVLA first predicts a deterministic low-frequency action anchor. It then uses a residual diffusion bridge to refine local motion details (Zhong et al., 2026).

FReD-VLA also starts refinement from an informative state, but studies how to connect a Flow generator to an existing diffusion action model. Its starting point is a *Flow-generated action sequence* conditioned on visual observations, language instructions, and robot states. Its refiner is a pretrained *diffusion action model*. ADC converts the coarse sequence into a noisy intermediate input. It uses the signal scaling and noise level specified by the diffusion schedule at the selected timestep. This allows refinement to start from the Flow proposal using the denoiser's own noising rule.

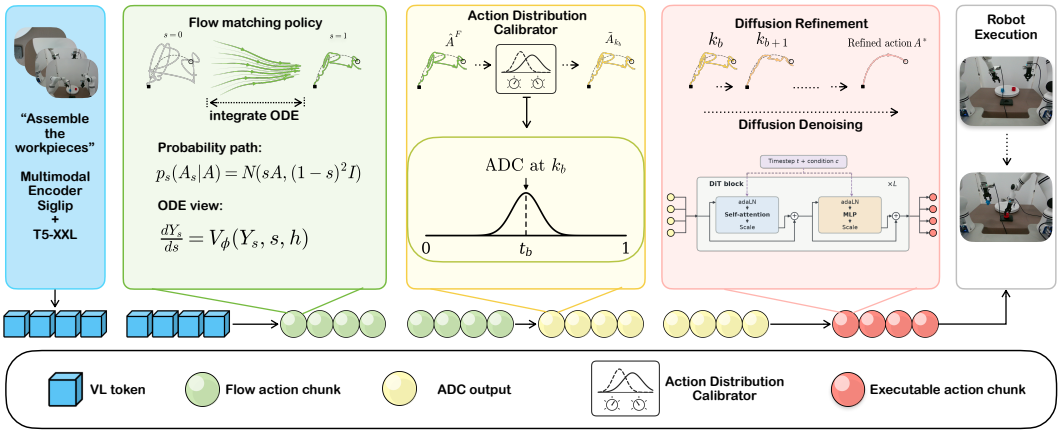


Figure 1: Overview of FReD-VLA. SigLIP and T5-XXL encode visual observations and language instructions. The Flow policy uses these features and the robot state to generate a coarse action chunk $\hat{\mathbf{A}}^F$. ADC applies the diffusion model's forward noise kernel at timestep k_b. A diffusion Transformer then refines the noisy chunk into the final actions $\hat{\mathbf{A}}$. The inset shows a DiT block with timestep and multimodal conditions.

3 Method

3.1 Problem formulation

We study language-conditioned bimanual manipulation. We use a small set of demonstrations to adapt the policy to a target task. At decision step t, the policy receives a task condition and predicts an action chunk:

$$\mathcal{C}_t = (\ell, \mathbf{X}_{t-T_v+1:t}, \mathbf{z}_t, f_c), \qquad \mathbf{A}_t = [\mathbf{a}_t, \dots, \mathbf{a}_{t+H-1}] \in \mathbb{R}^{H \times d_a}. \tag{1}$$

Here ℓ is the language instruction, $\mathbf{X}$ is a short history of multi-view RGB images, and $\mathbf{z}_t$ is the robot state. The control frequency is f_c. The chunk contains H actions, each with d_a valid action dimensions. A multimodal encoder produces the condition $\mathbf{h}_t = E(\mathcal{C}_t)$. We omit t below when the context is clear. Our goal is to retain the action quality of diffusion policies (Chi et al., 2023; Liu et al., 2025) while reducing their inference cost.

3.2 FReD-VLA: Flow-initialized diffusion refinement

FReD-VLA has three stages (Figure 1). Flow generates a coarse proposal $\hat{\mathbf{A}}^F$. The Action Distribution Calibrator (ADC) applies the diffusion forward kernel to obtain $\tilde{\mathbf{A}}_{k_b}$. Diffusion then refines this state into $\hat{\mathbf{A}}$ with a short denoising sequence. Both generators share the task condition $\mathbf{h}$.

3.3 Conditional Flow proposal

Let $\boldsymbol{\epsilon}_F \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ have the same shape as the target action chunk $\mathbf{A}$. We use a straight path from noise to the target:

$$\mathbf{A}_s = (1-s)\boldsymbol{\epsilon}_F + s\mathbf{A}, \qquad \mathbf{u}_s = \frac{d\mathbf{A}_s}{ds} = \mathbf{A} - \boldsymbol{\epsilon}_F, \quad s \in [0,1]. \tag{2}$$

We train the conditional velocity field $\mathbf{v}_\phi$ to predict $\mathbf{A} - \boldsymbol{\epsilon}_F$ (Lipman et al., 2023; Liu et al., 2023):

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\left[\|\mathbf{v}_\phi(\mathbf{A}_s, s, \mathbf{h}) - (\mathbf{A} - \boldsymbol{\epsilon}_F)\|_2^2\right]. \tag{3}$$

The expectation is over demonstration pairs, Gaussian noise, and $s \sim \mathcal{U}[0,1]$.

At inference, we solve $d\mathbf{Y}_s/ds = \mathbf{v}_\phi(\mathbf{Y}_s, s, \mathbf{h})$ from $\mathbf{Y}_0 = \boldsymbol{\epsilon}_F$. For an Euler grid $0 = s_0 < \cdots < s_{N_F} = 1$, this gives

$$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta s_j \, \mathbf{v}_\phi(\mathbf{Y}^{(j)}, s_j, \mathbf{h}), \qquad \Delta s_j = s_{j+1} - s_j. \tag{4}$$

One Euler step gives $\mathbf{Y}^{(1)} = \boldsymbol{\epsilon}_F + \mathbf{v}_\phi(\boldsymbol{\epsilon}_F, 0, \mathbf{h})$. Each step uses one network function evaluation (NFE). The proposal is $\hat{\mathbf{A}}^F = \Pi(\mathbf{Y}^{(N_F)})$, where Π applies any fixed action mask or clipping and is otherwise the identity.

3.4 Action Distribution Calibrator

The diffusion model is trained to denoise actions at known noise levels. ADC uses its forward noise kernel to place the Flow proposal at a selected level. Let K be the number of training timesteps and $\bar{\alpha}_k$ the cumulative noise coefficient. For an ADC fraction $\rho \in (0, 1)$, define

$$k_b = \lfloor \rho(K-1) \rfloor, \qquad \alpha_b = \sqrt{\bar{\alpha}_{k_b}}, \qquad \sigma_b = \sqrt{1 - \bar{\alpha}_{k_b}}. \tag{5}$$

Expanding the Euler proposal in Equation 4 gives the full Flow–ADC mapping:

$$\begin{aligned} \tilde{\mathbf{A}}_{k_b} &= \mathcal{C}_{k_b}(\boldsymbol{\epsilon}_F, \boldsymbol{\epsilon}_B; \mathbf{h}) \\ &= \alpha_b \, \Pi \left[\boldsymbol{\epsilon}_F + \sum_{j=0}^{N_F - 1} \Delta s_j \, \mathbf{v}_\phi(\mathbf{Y}^{(j)}, s_j, \mathbf{h}) \right] + \sigma_b \boldsymbol{\epsilon}_B, \qquad \boldsymbol{\epsilon}_B \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \tag{6}$$

The two noise samples are independent. This expression shows how the learned Flow path and the selected diffusion noise level jointly define the refinement input. With one Flow step, the sum contains only $\mathbf{v}_\phi(\boldsymbol{\epsilon}_F, 0, \mathbf{h})$. Other Flow solvers pass their numerical output through the same ADC map. ADC itself requires no network evaluation.

Conditioned on the proposal, Equation 6 uses exactly the corruption kernel associated with k_b. This aligns the noise scale with the denoiser's timestep input. The resulting distribution can still differ from the true training marginal because the proposal is approximate. For the usual decreasing $\bar{\alpha}_k$ schedule, a smaller ρ preserves more of the proposal. A larger ρ adds more noise and leaves more of the action structure to be recovered by diffusion.

ADC also affects temporal structure. Let D_H take differences between adjacent actions in a chunk, and assume $H \geq 2$. Define the mean squared temporal difference

$$S_2(\mathbf{A}) = \frac{\|D_H \mathbf{A}\|_F^2}{(H-1)d_a}. \tag{7}$$

Each adjacent difference of independent unit Gaussian noise has variance two. The cross term has zero expectation. Thus, on the valid action dimensions,

$$\mathbb{E}_{\boldsymbol{\epsilon}_B}\left[S_2(\tilde{\mathbf{A}}_{k_b}) \mid \hat{\mathbf{A}}^F \right] = \alpha_b^2 S_2(\hat{\mathbf{A}}^F) + 2\sigma_b^2. \tag{8}$$

The two terms separate variation retained from Flow and variation added by ADC noise. This identity describes the refinement input. Here S_2 measures squared differences, whereas evaluation uses absolute-difference Step Diff. Final motion quality also depends on agreement with the target sequence. Section 3.6 studies errors in both action values and temporal increments after refinement.

3.5 Local diffusion refinement

Following RDT (Liu et al., 2025), f_θ predicts clean actions from noisy inputs. Appendix B.1, Equation 34, gives the standard training objective. The main offline and real-robot tests combine separately trained Flow and diffusion checkpoints without further updates. Simulation ablations also train the refiner while keeping Flow fixed.

Starting from the ADC output, the solver performs N_D reverse updates:

$$\begin{aligned} \mathbf{A}^{(j+1)} &= \mathcal{R}_\theta\Big(\mathbf{A}^{(j)}, k^{(j)}, k^{(j+1)}, \mathbf{h}\Big), \\ \mathbf{A}^{(0)} &= \tilde{\mathbf{A}}_{k_b}, \qquad k^{(0)} = k_b, \qquad \hat{\mathbf{A}} = \mathbf{A}^{(N_D)}. \end{aligned} \tag{9}$$

Here $\mathcal{R}_\theta$ is the backbone's reverse solver. With one denoiser call per update, N_D updates use N_D diffusion NFEs. Otherwise, we count the actual calls.

Let c_F and c_D be the costs of one Flow and one diffusion evaluation. Generation costs approximately $N_F c_F + N_D c_D$ plus ADC computation. Pure diffusion with N_P calls costs $N_P c_D$. FReD-VLA is cheaper when the saved diffusion work exceeds the added Flow and ADC work.

3.6 THEORETICAL ANALYSIS OF FEW-STEP REFINEMENT

We now relate the diffusion budget in Section 3.5 to a fixed quality target. In the first-order analysis below, write $N = N_D$. Following warm-start diffusion (Chung et al., 2022), we use a local Gaussian model and an ideal denoiser. Appendix A provides the full derivation.

Action quality and assumptions. We assess motion by comparing action values and temporal increments with the target. Vectorize an action chunk into $\mathbf{a} \in \mathbb{R}^n$, with $n = Hd_a$. The operator $D = D_H \otimes I_{d_a}$ extends the temporal differences in Section 3.4 to this vector, with $m = (H-1)d_a$ outputs. For $\lambda > 0$, define

$$Q_\lambda = \frac{I_n}{n} + \frac{\lambda}{m} D^\top D, \qquad \|\mathbf{e}\|_{Q_\lambda}^2 = \frac{\|\mathbf{e}\|_2^2}{n} + \lambda \frac{\|D\mathbf{e}\|_2^2}{m}. \tag{10}$$

For $\mathbf{e} = \hat{\mathbf{a}} - \mathbf{a}^\star$, the two terms measure errors in action values and reference temporal increments. Thus, an inaccurate constant trajectory can still have a large error. The metric adds no training loss. We use W_{2,Q_λ} for the Wasserstein distance with squared cost $\|\cdot\|_{Q_\lambda}^2$.

For a fixed condition, assume a local Gaussian target $p_0 = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Let $\Sigma = U \operatorname{diag}(v_q) U^\top$, with $v_q > 0$. Assume U also diagonalizes $D^\top D$, so $Q_\lambda = U \operatorname{diag}(w_q) U^\top$. We use the exact conditional-mean denoiser and deterministic first-order DDIM/DPM-Solver++ updates (Song et al., 2020; Lu et al., 2022). These assumptions allow each mode to be analyzed separately.

Separating proposal and sampling errors. To isolate sampling error, we first replace the Flow proposal with an exact sample from p_0. ADC sets the starting noise-to-signal ratio to $r_b = \sigma_b/\alpha_b$. Consider N reverse steps on $r_0 = r_b > \cdots > r_N = 0$, with j indexing these steps. Define $\vartheta_{j,q} = \arctan(r_j/\sqrt{v_q})$ and $\Delta\vartheta_{j,q} = \vartheta_{j,q} - \vartheta_{j+1,q}$. The updates preserve the mean and give output variance $v_q c_{q,N}^2$. Appendix A, Equation 22, derives this result. The variance factor and sampling error are

$$c_{q,N} = \prod_{j=0}^{N-1} \cos(\Delta\vartheta_{j,q}), \qquad E_{\mathrm{disc}}(b,N) = \left[\sum_q w_q v_q (1 - c_{q,N})^2 \right]^{1/2}. \tag{11}$$

This follows the Gaussian sampling-angle analysis of Sabour et al. (2024). Finite steps shrink the target variance. The resulting sampling error E_{disc} decreases as the grid is refined.

We next include the error of the actual Flow proposal. Let p_F have finite second moment and proposal error $\delta_F = W_{2,Q_\lambda}(p_F, p_0)$. Couple Flow and target samples optimally and add the same independent ADC noise. The denoising map contracts their difference, as shown in Equation 24 of Appendix A. Thus,

$$W_{2,Q_\lambda}(p_{b,N}^F, p_0) \le \Gamma_b \delta_F + E_{\mathrm{disc}}(b,N), \qquad \Gamma_b = \max_q \sqrt{\frac{v_q}{v_q + r_b^2}}. \tag{12}$$

Here $p_{b,N}^F$ is the refined distribution. The bound separates retained Flow error from finite-step sampling error.

From error to denoising steps. Write $\vartheta_{b,q} = \vartheta_{0,q}$. Assume $N \sum_j (\Delta\vartheta_{j,q})^2 \le \kappa_b \vartheta_{b,q}^2$ across the modes and budgets considered, with a common $\kappa_b \ge 1$. This limits uneven angular steps. Since $1 - c_{q,N} \le \frac{1}{2} \sum_j (\Delta\vartheta_{j,q})^2$, substitution yields the following result.

Proposition 1 (Initialization error and a finite denoising budget)**.** *Under the stated model, denoiser, and grid assumptions,*

$$W_{2,Q_\lambda}(p_{b,N}^F, p_0) \le \Gamma_b \delta_F + \frac{B_b}{N}, \qquad B_b = \frac{\kappa_b}{2} \left[\sum_q w_q v_q \vartheta_{b,q}^4 \right]^{1/2}. \tag{13}$$

For a tolerance $\epsilon > \Gamma_b \delta_F$, solving this inequality gives the sufficient budget

$$N_{\mathrm{cert}}(b,\epsilon) = \max\left\{ 1, \left\lceil \frac{B_b}{\epsilon - \Gamma_b \delta_F} \right\rceil \right\}. \tag{14}$$

Table 1: Offline action quality and latency. Lower is better for all four metrics. Latency excludes shared multimodal encoding. Flow / Diff. NFE lists the Flow and diffusion evaluation counts; ρ is the ADC fraction. The radar plot excludes Pure Flow Euler-1.

Method	Flow / Diff. NFE	ρ	Joint MAE (°) ↓	Sample Std ↓	Step Diff ↓	Latency ms ↓
Pure Diffusion-2	0 / 2	–	0.5758	0.1992	0.1695	54.1
Pure Diffusion-5	0 / 5	–	0.5892	0.2053	0.1665	151.9
Pure Flow Euler-1	1 / 0	–	1.1760	1.2833	0.4522	**4.6**
Pure Flow Euler-8	8 / 0	–	0.8214	0.7717	0.2549	37.5
XVLA	–	–	2.0033	0.3832	0.2153	72.0
DexVLA	–	–	8.9777	4.8620	0.4741	86.2
SmolVLA	–	–	**0.1817**	0.2321	**0.1565**	203.2
FReDVLA (ours)	1 / 1	0.15	0.6201	**0.1932**	0.1640	37.0

Better proposals leave more tolerance for sampling error and can reduce this budget. One step suffices if $\Gamma_b \delta_F + B_b \leq \epsilon$. Refinement improves the proposal when $B_b/N < (1 - \Gamma_b)\delta_F$. For comparable grid regularity, a lower-noise start reduces B_b but increases retained proposal error through Γ_b. A sufficiently accurate proposal can therefore support a shorter denoising interval at the chosen tolerance.

Appendix A, Equation 31, also gives a strict scalar comparison. The baseline uses a higher-noise start from its exact forward marginal. When the hybrid's sufficient budget is below the baseline's necessary budget, the hybrid reaches the same tolerance with strictly fewer denoiser evaluations. The distance here measures distributional fidelity, not paired Joint MAE. Learned-denoiser error and total latency, including Flow computation, require empirical evaluation.

Under these assumptions, the analysis connects proposal quality to a sufficient denoising budget. It gives conditions for fewer refinement steps and for improving the proposal. This explains how an accurate Flow estimate can reduce diffusion computation while preserving action values and temporal structure.

4 Experiments

We address three research questions: **(Q1)** whether short diffusion refinement can recover action fidelity from a coarse Flow proposal while reducing repeated denoising; **(Q2)** how Flow NFE and the ADC fraction allocate computation between transport and correction; and **(Q3)** whether FReDVLA supports closed-loop simulation and latency-sensitive real-robot execution. We evaluate FReDVLA through offline action prediction, ManiSkill, and dynamic bimanual assembly on a real robot. Appendix B.1 provides the full setup.

4.1 Action accuracy–latency trade-off

Table 1 reports the raw results. Figure 2 summarizes seven methods, with normalization defined in its caption and detailed in Appendix B.2.

One diffusion evaluation reduces Euler-1 Joint MAE from $1.1760°$ to $0.6201°$. The total generation time is 37.0 ms. Euler-8 takes a similar 37.5 ms but reaches $0.8214°$. FReDVLA also has lower Sample Std and Step Diff than Euler-8. Thus, diffusion correction uses this budget more effectively than additional Flow steps.

FReDVLA is $4.1\times$ faster than Pure Diffusion-5 (37.0 versus 151.9 ms), with close Joint MAE ($0.6201°$ versus $0.5892°$). Pure Diffusion-2 has lower Joint MAE ($0.5758°$) but takes 54.1 ms. FReDVLA therefore trades a small increase in joint error for lower latency.

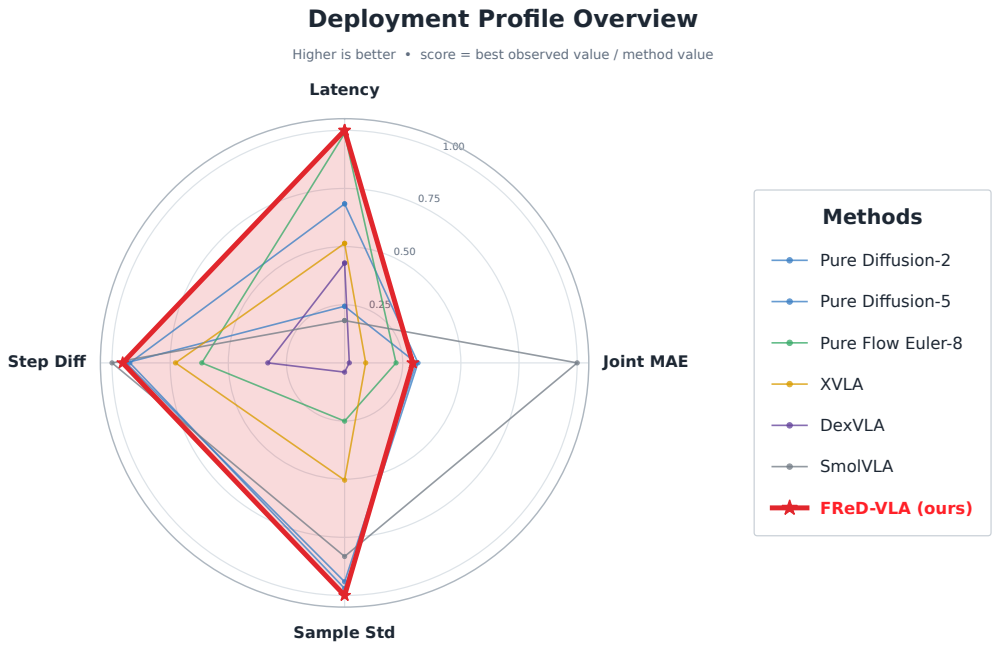


Figure 2: Latency and action quality for seven methods from Table 1, excluding Pure Flow Euler-1. Each score is the lowest plotted value divided by the method's value. Higher is better. Latency excludes shared multimodal encoding.

Table 2: Representative Flow solver/NFE sweep.

Flow solver	NFE	Joint MAE ↓	Sample Std ↓	Step Diff ↓	ms ↓
Euler-1	1	1.1760	1.2833	0.4522	**4.6**
Euler-8	8	0.8214	**0.7717**	**0.2549**	37.5
Euler-16	16	**0.8175**	0.8304	0.2555	80.9
RK4-80	320	0.8319	1.0105	0.2800	1483.9

SmolVLA (Shukor et al., 2025) has the lowest Joint MAE (0.1817°) but takes 203.2 ms. XVLA (Zheng et al., 2025) and DexVLA (Wen et al., 2025) have higher Joint MAE and latency than FReDVLA. FReDVLA has the lowest Sample Std in the table.

The 37.0 ms generation time is below the 40 ms sampling interval at 25 Hz. This comparison covers action generation and excludes shared multimodal encoding.

4.2 How much Flow transport is needed?

Table 2 shows that Flow quality saturates quickly. Euler-8 reaches 0.8214° in 37.5 ms, whereas RK4-80 uses 320 vector-field evaluations and 1483.9 ms for 0.8319°. More importantly, after one diffusion correction, Euler-1 and Euler-8 proposals reach 0.6201° and 0.6193°, respectively, while the latter nearly doubles latency (37.0 versus 73.8 ms). FReDVLA therefore benefits from *stopping Flow early*. The complete sweep is in Appendix B.3.

4.3 Effect of the ADC fraction

Table 3 varies ρ with one Flow and one diffusion evaluation. Joint MAE improves from $\rho = 0.05$ to an intermediate optimum at 0.15, then worsens as stronger corruption removes more proposal structure. The same optimum also appears with an eight-step Euler proposal (Appendix B.4), showing that it is not specific to the noisiest one-step Flow sample.

Table 3: ADC fraction ablation with one Flow NFE and one diffusion NFE.

Calib ρ	Joint MAE ↓	Sample Std ↓	Step Diff ↓	Latency ms ↓	Refinement gain ↑
0.05	0.6478	**0.1455**	**0.1630**	32.6	44.92%
0.10	0.6262	0.1681	0.1636	38.7	46.75%
0.15	**0.6201**	0.1932	0.1640	37.0	**47.27%**
0.20	0.6244	0.2219	0.1656	**30.9**	46.91%
0.30	0.6595	0.2837	0.1699	40.2	43.92%



Figure 3: Dynamic real-robot bimanual assembly. From left to right, the robot acquires moving workpieces from a rotary feeder, transfers them to the assembly area, and assemble the parts.

4.4 Controlled closed-loop simulation

On ManiSkill PickCube-v1, Pure Flow and Flow-conditioned Step50 each succeed in $3/9$ rollouts, compared with $2/9$ for Pure Diffusion and Standard500. Standard500 and Flow-conditioned Step50 both achieve first grasp in $3/9$ cases. Flow-conditioned Step50 retains the Pure Flow mean minimum TCP–cube distance of 0.0108 ± 0.0229 m, compared with 0.0513 ± 0.0229 m for Standard500 and 0.0437 ± 0.0702 m for Pure Diffusion. Frozen achieves neither success nor first grasp ($0/9$ for both). Under this fixed untuned seed grid, Flow-conditioned Step50 preserves the stronger closed-loop behavior of the Flow proposal while retaining diffusion refinement. Full matched statistics and heterogeneous external-policy transfer references are reported in Appendix B.5–B.6.

4.5 Dynamic real-robot bimanual assembly

We test FReDVLA on dynamic bimanual assembly (Figure 3). A rotary feeder moves two workpiece types. One arm holds the primary part, while the other picks the mating part and completes assembly. Each behavior is adapted using 20–30 human demonstrations. Inference uses one Flow and one diffusion evaluation. The demonstration verifies integration of the Flow proposal, ADC, diffusion refinement, and bimanual execution. Section 4.1 provides accuracy–latency comparison.

5 Conclusion

We presented FReDVLA to balance action accuracy and generation speed in bimanual manipulation with limited demonstrations. Flow matching generates a coarse action sequence. Our Action Distribution Calibrator (ADC) maps this proposal to an intermediate noisy state. The pretrained diffusion model then refines this state into the final actions.

In offline evaluation, one diffusion step reduces the Joint MAE of a one-step Flow proposal from $1.1760°$ to $0.6201°$. With one Flow evaluation and one diffusion evaluation, FReDVLA takes 37.0 ms, excluding shared multimodal encoding. This gives a $4.1\times$ speedup over five-step diffusion at 151.9 ms, with similar action accuracy. Solver and ADC fraction ablations, closed-loop simulation, and dynamic real-robot assembly further support the approach. These results show the value of using Flow for fast proposals and Diffusion for refinement within one policy.

AI use statement

Generative AI tools were used for drafting and language-editing assistance and for iterative design assistance on the method schematic. All technical claims, mathematical derivations, experimental procedures, reported results, and final manuscript content were reviewed and verified by the authors.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2410.24164.

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

Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. *arXiv preprint arXiv:2112.05146*, 2022. URL https://arxiv.org/abs/2112.05146.

Zhenchen Dong, Jinna Fu, Jiaming Wu, Shengyuan Yu, Fulin Chen, and Yide Liu. Hybridflow: A two-step generative policy for robotic manipulation. *arXiv preprint arXiv:2602.13718*, 2026.

Dechen Gao, Boqi Zhao, Andrew Lee, Ian Chuang, Hanchu Zhou, Hang Wang, Zhe Zhao, Junshan Zhang, and Iman Soltani. Vita: Vision-to-action flow matching policy. In *International Conference on Learning Representations*, 2026.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. In *International Conference on Learning Representations*, 2025.

Jiyuan Jing, Aoling Yang, and Fang Peng. Mix diffusion policy: hierarchical robotic manipulation via diffusion-flow matching fusion. In *Proceedings of SPIE*, volume 14253, pp. 142530I. SPIE, 2026. doi: 10.1117/12.3117241. URL https://doi.org/10.1117/12.3117241.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

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. In *International Conference on Learning Representations*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. URL https://arxiv.org/abs/2211.01095.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. *arXiv preprint arXiv:2404.14507*, 2024. URL https://arxiv.org/abs/2404.14507.

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. URL https://arxiv.org/abs/2506.01844.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. URL https://arxiv.org/abs/2010.02502.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025. URL https://arxiv.org/abs/2502.05855.

Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Efficient diffusion model for image restoration by residual shifting. *arXiv preprint arXiv:2403.07319*, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025. URL https://arxiv.org/abs/2510.10274.

Yiming Zhong, Yaoyu He, Zemin Yang, Pengfei Tian, Yifan Huang, Qingqiu Huang, Xinge Zhu, and Yuexin Ma. From noise to intent: Anchoring generative vla policies with residual bridges. *arXiv preprint arXiv:2604.21391*, 2026.

A Temporal-structure analysis of few-step refinement

This appendix proves Proposition 1 and derives a sufficient denoising budget. The analysis is exact for the stated conditional Gaussian model and deterministic first-order sampler. Its purpose is to connect Flow initialization, temporal structure, and finite sampling computation. Gaussian sampling-angle analysis has precedent in Sabour et al. (2024); we combine that analysis with imperfect proposal initialization and a temporal-difference transport metric.

A.1 Assumptions and the temporal metric

Fix the multimodal condition and vectorize the valid action dimensions into $\mathbf{a} \in \mathbb{R}^n$, where $n = Hd_a$. Let $D = D_H \otimes I_{d_a}$ be the first-difference matrix, with $m = (H-1)d_a$ rows. Assume $H \geq 2$ and fix $\lambda > 0$. Then

$$Q = Q_\lambda = I_n/n + \lambda D^\top D/m \succ 0, \qquad W_{2,Q}(P, P') = \inf_{\pi \in \Pi(P,P')} \left(\mathbb{E}_\pi \|\mathbf{a} - \mathbf{a}'\|_Q^2\right)^{1/2}. \quad (15)$$

All distributions below have finite second moments. We assume

$$p_0 = \mathcal{N}(\boldsymbol{\mu}, \Sigma), \quad \Sigma = U \operatorname{diag}(v_q) U^\top, \quad D^\top D = U \operatorname{diag}(\nu_q) U^\top, \quad v_q > 0. \quad (16)$$

Thus $Q = U \operatorname{diag}(w_q) U^\top$, with $w_q = 1/n + \lambda \nu_q / m$. Simultaneous diagonalization is an explicit surrogate assumption. It need not hold for a nonlinear, multimodal action policy. The mean $\boldsymbol{\mu}$ is a conditional mean trajectory and is not assumed to equal a particular expert trajectory.

A.2 The exact denoiser and first-order update

For a diffusion state $\mathbf{x}_k$, introduce

$$r_k = \sqrt{\frac{1 - \bar{\alpha}_k}{\bar{\alpha}_k}}, \qquad \mathbf{y}_k = \frac{\mathbf{x}_k}{\sqrt{\bar{\alpha}_k}}. \quad (17)$$

The forward observation is $\mathbf{y}_k = \mathbf{a}_0 + r_k\boldsymbol{\epsilon}$, while the ADC initialization is $\mathbf{y}_b = \mathbf{a}_F + r_b\boldsymbol{\epsilon}$. The Gaussian noise is independent of the clean action or Flow proposal. Under Equation 16, Gaussian conditioning gives the squared-loss-optimal clean-action denoiser

$$f^*(\mathbf{y}, r) = \boldsymbol{\mu} + \Sigma(\Sigma + r^2 I)^{-1}(\mathbf{y} - \boldsymbol{\mu}). \tag{18}$$

Consider a grid $r_0 = r_b > r_1 > \cdots > r_N = 0$, where the index zero denotes the start of reverse sampling. Deterministic DDIM in these coordinates is

$$\mathbf{y}_{j+1} = q_j\mathbf{y}_j + (1 - q_j)f^*(\mathbf{y}_j, r_j), \qquad q_j = r_{j+1}/r_j. \tag{19}$$

This is also the deterministic first-order DPM-Solver++ update (Song et al., 2020; Lu et al., 2022). We use the ideal clean endpoint $r_N = 0$; a nonzero endpoint or additional postprocessing requires its own error term.

For $z_{j,q} = \mathbf{u}_q^\top(\mathbf{y}_j - \boldsymbol{\mu})$, substitution yields

$$z_{j+1,q} = g_{j,q}z_{j,q}, \qquad g_{j,q} = \frac{v_q + r_j r_{j+1}}{v_q + r_j^2}. \tag{20}$$

Define $\vartheta_{j,q} = \arctan(r_j/\sqrt{v_q})$ and $\Delta\vartheta_{j,q} = \vartheta_{j,q} - \vartheta_{j+1,q} \geq 0$. The cosine difference identity gives

$$g_{j,q} = \sqrt{\frac{v_q + r_{j+1}^2}{v_q + r_j^2}}\cos(\Delta\vartheta_{j,q}). \tag{21}$$

Multiplying over the grid cancels the intermediate square-root factors:

$$z_{N,q} = \gamma_{b,q}c_{q,N}z_{0,q}, \qquad \gamma_{b,q} = \sqrt{\frac{v_q}{v_q + r_b^2}}, \qquad c_{q,N} = \prod_{j=0}^{N-1}\cos(\Delta\vartheta_{j,q}). \tag{22}$$

The continuous probability-flow equation is $dz_q/dr = rz_q/(v_q + r^2)$ and has endpoint $\gamma_{b,q}z_{0,q}$. Therefore $c_{q,N}$ is the additional attenuation caused by finite-step discretization. Equation 22 is valid also for $N = 1$; it is not an asymptotic extrapolation from large N.

A.3 Proof of the finite-budget bound

First initialize with $\mathbf{a}_0 \sim p_0$. The initial modal variance is $v_q + r_b^2$, and Equation 22 gives output variance $v_q c_{q,N}^2$, with unchanged mean. Because the covariance matrices and Q share the eigenbasis U, the Gaussian transport formula gives

$$E_{\mathrm{disc}}(b, N) = \left[\sum_q w_q v_q (1 - c_{q,N})^2\right]^{1/2}. \tag{23}$$

This can also be seen by coupling corresponding modes with the same standard Gaussian: their standard deviations differ by $\sqrt{v_q}(1 - c_{q,N})$.

Now let $\mathbf{a}_F \sim p_F$ and couple it optimally with $\mathbf{a}_0 \sim p_0$ under Q. Apply the same independent ADC noise to both initializations. Their initial difference is $\mathbf{a}_F - \mathbf{a}_0$. After N steps it becomes

$$U\,\mathrm{diag}(\gamma_{b,q}c_{q,N})U^\top(\mathbf{a}_F - \mathbf{a}_0). \tag{24}$$

Since $0 \leq c_{q,N} \leq 1$, its Q-norm mean square is at most $\Gamma_b^2\delta_F^2$, where $\Gamma_b = \max_q \gamma_{b,q}$ and $\delta_F = W_{2,Q}(p_F, p_0)$. The triangle inequality then gives

$$W_{2,Q}(p_{b,N}^F, p_0) \leq \Gamma_b\delta_F + E_{\mathrm{disc}}(b, N). \tag{25}$$

No independence between the proposal error and the target action is needed. A specified, nonoptimal coupling gives a valid but potentially looser upper bound on δ_F.

For any nonnegative angular increments summing to $\vartheta_{b,q}$,

$$1 - c_{q,N} \leq \sum_j \left[1 - \cos(\Delta\vartheta_{j,q})\right] \leq \frac{1}{2}\sum_j(\Delta\vartheta_{j,q})^2. \tag{26}$$

Under the grid condition in Proposition 1, this is at most $\kappa_b \vartheta_{b,q}^2/(2N)$. Substituting into Equation 23 yields

$$E_{\mathrm{disc}}(b, N) \le \frac{B_b}{N}, \qquad B_b = \frac{\kappa_b}{2}\left[\sum_q w_q v_q \vartheta_{b,q}^4\right]^{1/2}, \tag{27}$$

which proves Proposition 1. The common κ_b must bound all grids to which the budget claim is applied; it is not automatically independent of N. Without this regularity assumption, Equation 25 remains valid for each specified grid.

A.4 Fidelity improvement and step-count comparison

Solving $\Gamma_b \delta_F + B_b/N \le \epsilon$ for an integer $N \ge 1$ gives Equation 14, provided $\epsilon > \Gamma_b \delta_F$. In addition,

$$\frac{B_b}{N} < (1 - \Gamma_b)\delta_F \quad \Longrightarrow \quad W_{2,Q}(p_{b,N}^F, p_0) < W_{2,Q}(p_F, p_0). \tag{28}$$

Thus the same bound gives conditions for both a prescribed fidelity tolerance and improvement over the proposal. The latter comparison uses the actual δ_F, not merely a loose upper bound on it.

A scalar mode permits a sharper comparison. Let its variance and metric weight be v and w, and let its starting angle be ϑ. Concavity of $\log\cos x$ implies

$$c_N \le \cos(\vartheta/N)^N, \tag{29}$$

with equality for equal angular increments. This recovers the Gaussian optimal-angle schedule of Sabour et al. (2024). For a matched initialization, the minimum attainable relative distribution error at budget N is therefore $1 - \cos(\vartheta/N)^N$. Let $N_{\min}(\vartheta, \eta)$ be the smallest budget achieving relative error at most $\eta \in (0, 1)$. Using $-\log\cos x \ge x^2/2$ and $1 - \cos(\vartheta/N)^N \le \vartheta^2/(2N)$ yields

$$\begin{aligned} \max\left\{1, \left\lceil \frac{\vartheta^2}{2[-\log(1-\eta)]} \right\rceil\right\} &\le N_{\min}(\vartheta, \eta) \\ &\le \max\left\{1, \left\lceil \frac{\vartheta^2}{2\eta} \right\rceil\right\}. \end{aligned} \tag{30}$$

The relative error is increasing in ϑ for fixed N. Consequently, the optimal scalar budget is nondecreasing in the starting angle, although integer rounding can make two budgets equal.

For an imperfect Flow initialization at angle ϑ_b, let $\gamma_b = \cos\vartheta_b$ and choose a tolerance with $\eta_H = (\epsilon - \gamma_b\delta_F)/\sqrt{wv} \in (0, 1)$. Compare this with a higher-noise sampler at angle $\vartheta_T > \vartheta_b$, initialized from its exact forward marginal, and set $\eta_D = \epsilon/\sqrt{wv} \in (0, 1)$. An upper bound for the hybrid budget and a lower bound for the higher-noise budget are

$$\begin{aligned} N_H^{\mathrm{up}} &= \max\left\{1, \left\lceil \frac{\vartheta_b^2}{2\eta_H} \right\rceil\right\}, \\ N_D^{\mathrm{low}} &= \max\left\{1, \left\lceil \frac{\vartheta_T^2}{2[-\log(1-\eta_D)]} \right\rceil\right\}. \end{aligned} \tag{31}$$

If $N_H^{\mathrm{up}} < N_D^{\mathrm{low}}$, the hybrid's equal-angle schedule attains the tolerance with fewer evaluations than the higher-noise sampler requires even with its optimal grid. This is a strict step-count result within the scalar idealization, not a comparison of two sufficient upper bounds. Pure-noise initialization is represented by the limit $r_T \to \infty$, or $\vartheta_T \to \pi/2$. At finite terminal noise, a standard Gaussian initialization may differ from the exact forward marginal and needs an additional initialization-error term.

For a fixed tolerance and grid regularity, smaller δ_F increases the admissible error budget for discretization and can permit a smaller r_b. This is the mechanism by which an informative proposal can reduce denoising computation. It does not prescribe a universally optimal ADC fraction.

A.5 Learned denoisers and interpretation of the metric

For completeness, compare a learned first-order sampler with the ideal one on the same initialization and grid. Let ξ_j be the Q-norm root-mean-square discrepancy between its clean-action prediction

and f^*, evaluated along the learned path. Writing $\gamma_j = \max_q g_{j,q} \le 1$, the affine ideal update and Minkowski's inequality give the output discrepancy bound

$$\varepsilon_\theta(b, N) \le \sum_{j=0}^{N-1} (1 - q_j)\xi_j \prod_{\ell=j+1}^{N-1} \gamma_\ell. \tag{32}$$

Hence $W_{2,Q}(p_{b,N}^{F,\theta}, p_0) \le \Gamma_b \delta_F + B_b/N + \varepsilon_\theta(b, N)$. A budget for the learned model would require an independently justified bound on this additional term. The ideal proposition does not set it to zero for the actual policy. High-order or stochastic samplers require their own update analysis.

The temporal metric controls fidelity of motion increments rather than forcing motion to vanish. For $\mathrm{SD}_1(\mathbf{a}) = \|D\mathbf{a}\|_1/m$ and a fixed reference action $\mathbf{a}_{\mathrm{ref}}$, Cauchy–Schwarz gives

$$\begin{aligned} |\,\mathrm{SD}_1(\mathbf{a}) - \mathrm{SD}_1(\mathbf{a}')| &\le \|\mathbf{a} - \mathbf{a}'\|_Q/\sqrt{\lambda}, \\ |\,\mathrm{MAE}(\mathbf{a}, \mathbf{a}_{\mathrm{ref}}) - \mathrm{MAE}(\mathbf{a}', \mathbf{a}_{\mathrm{ref}})| &\le \|\mathbf{a} - \mathbf{a}'\|_Q. \end{aligned} \tag{33}$$

Accordingly, $W_{2,Q}$ bounds deviations of these expected metrics from those of p_0. All quantities here use the same action coordinates; joint-angle units require the corresponding inverse-normalization factors. The equations do not identify distributional distance with the empirical Joint MAE, prove improvement for every expert trajectory, or imply closed-loop success. They also do not imply that smaller raw Step Diff is always desirable: finite-step variance collapse can reduce this metric. The experimental metrics and latency remain empirical evidence, with Flow computation included in the cost.

B Additional Experimental Details and Results

B.1 Experimental setup details

Models and data. We use about 20–30 unique human demonstrations per real-robot behavior. Pure Diffusion, Pure Flow, and FReDVLA share the same DiT multimodal backbone. FReDVLA combines the separately trained baseline checkpoints without further weight updates. The main offline and real-robot evaluations use the same FReDVLA model version and settings. Our default is Euler-1, $\rho = 0.15$, and one diffusion evaluation.

Diffusion training objective. Following RDT (Liu et al., 2025), the diffusion backbone predicts the clean expert action chunk from a noisy input:

$$\begin{aligned} \tilde{\mathbf{A}}_k &= \sqrt{\bar{\alpha}_k}\,\mathbf{A} + \sqrt{1 - \bar{\alpha}_k}\,\boldsymbol{\epsilon}, \\ \mathcal{L}_{\mathrm{diff}} &= \mathbb{E}\left[\left\|\mathbf{A} - f_\theta(\tilde{\mathbf{A}}_k, k, \mathbf{h})\right\|_2^2\right], \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \tag{34}$$

The expectation is over expert action–condition pairs, training timesteps, and independent Gaussian noise. This is the standard diffusion objective used in Section 3.5. The refiner adaptation settings for simulation are described in Appendix B.5.

Offline evaluation. We evaluate nine trajectory anchors against recorded expert actions. Ten sampling seeds per anchor give 90 predictions per method. Conditions, action masks, and target chunks are identical across methods. We report Joint MAE on active arm joints, Sample Std across predictions, and Step Diff between adjacent actions. Latency is measured on one RTX 5090 with 10 warm-up calls and 30 synchronized timed calls. It excludes shared multimodal encoding. Timing includes Flow's context adaptor and integration, plus ADC and diffusion refinement for FReDVLA.

Simulation and hardware. We evaluate ManiSkill PickCube-v1 on a fixed 3×3 grid of sampling and environment seeds. Diffusion and Flow are trained separately on official ManiSkill data for 5,000 steps each using two RTX 5090 GPUs. Refiner ablations share one Flow checkpoint, Euler-1, $\rho = 0.2$, and one refinement step. Checkpoints and adaptation details are in Appendix B.5. External-policy comparisons are in Appendix B.6. Real-robot tests use a rotary feeder and two workpiece types for dynamic bimanual assembly. We use the same expert-derived residual execution layer throughout deployment. Offline generator metrics are measured before this layer.

B.2 Offline metrics and radar normalization

Table 1 in Section 4.1 reports the raw offline action-quality and latency metrics. All methods use the evaluation protocol in Section B.1. Lower values indicate lower action error, sampling variation, changes between adjacent actions, or generation latency. Step Diff is read together with Joint MAE, since a nearly static action sequence can also have a small Step Diff.

Figure 2 shows seven methods from this table. For each axis, we divide the lowest raw value among these seven methods by the value for the method being plotted. A value of 1 marks the best result on that axis. The reference values are 37.0 ms for latency, $0.1817°$ for Joint MAE, 0.1932 for Sample Std, and 0.1565 for Step Diff. Each axis is read separately. Pure Flow Euler-1 is retained in the table and the Flow NFE sweep; it is not part of the radar normalization.

B.3 Complete Flow solver and NFE sweep

Table 4 reports the complete Flow-only solver sweep used to select the low-NFE proposal regimes. Each entry is evaluated on the same 9 trajectory anchors $\times$ 10 stochastic sampling seeds as the main offline experiment.

Table 4: Complete Flow solver/NFE sweep. Lower is better.

Method	Solver	NFE	Mean ms
Euler-1	Euler	1	4.615
Euler-2	Euler	2	9.700
Euler-4	Euler	4	18.521
Euler-8	Euler	8	37.456
Euler-16	Euler	16	80.925
Euler-32	Euler	32	163.082
Euler-80	Euler	80	385.702
RK4-1	RK4	4	21.561
RK4-2	RK4	8	44.462
RK4-4	RK4	16	74.968
RK4-8	RK4	32	164.700
RK4-16	RK4	64	307.176
RK4-32	RK4	128	621.672
RK4-80	RK4	320	1483.942

Method	Joint MAE	Sample Std	Step Diff
Euler-1	1.176025	1.283288	0.452223
Euler-2	0.948387	0.810592	0.314598
Euler-4	0.849418	0.717700	0.263880
Euler-8	0.821397	0.771707	0.254895
Euler-16	0.817526	0.830383	0.255485
Euler-32	0.818938	0.905009	0.263830
Euler-80	0.836177	1.049806	0.291436
RK4-1	3.585525	10.696380	4.368685
RK4-2	1.677117	4.488284	1.774951
RK4-4	1.065594	2.243412	0.851794
RK4-8	0.873148	1.344750	0.470059
RK4-16	0.824693	1.024727	0.316059
RK4-32	**0.817201**	0.974722	0.274324
RK4-80	0.831869	1.010514	0.280023

B.4 Complete low-NFE sweep over the ADC fraction

Table 5 reports the full low-NFE FReDVLA sweep. Both Euler-1 and Euler-8 proposals achieve their lowest Joint MAE at $\rho = 0.15$.

Table 5: Complete low-NFE FReDVLA sweep. Each FReDVLA row uses one denoiser call.

Flow	ρ	Joint MAE	First MAE	Gain
Euler-1	0.05	0.647760	0.837790	44.92%
Euler-1	0.10	0.626215	0.773204	46.75%
Euler-1	0.15	**0.620131**	0.759443	**47.27%**
Euler-1	0.20	0.624395	0.767116	46.91%
Euler-1	0.30	0.659485	0.810699	43.92%
Euler-8	0.05	0.648429	0.830167	21.06%
Euler-8	0.10	0.628117	0.777494	23.53%
Euler-8	0.15	**0.619316**	0.756945	**24.60%**
Euler-8	0.20	0.623432	0.761429	24.10%
Euler-8	0.30	0.655693	0.811529	20.17%

Flow	ρ	Sample Std	Step Diff	Mean ms
Euler-1	0.05	0.145475	0.163002	32.614
Euler-1	0.10	0.168136	0.163591	38.719
Euler-1	0.15	0.193229	0.163994	37.020
Euler-1	0.20	0.221856	0.165581	30.876
Euler-1	0.30	0.283698	0.169869	40.152
Euler-8	0.05	0.155310	0.164216	65.521
Euler-8	0.10	0.175987	0.163731	71.253
Euler-8	0.15	0.199528	0.164365	73.806
Euler-8	0.20	0.229610	0.166034	66.983
Euler-8	0.30	0.287323	0.170640	80.648

B.5 ManiSkill matched internal comparison

All rows in Table 6 use the same sampling seeds $\{01, 02, 03\}$ crossed with environment seeds $\{20241201, 20241202, 20241203\}$. Because these models are evaluated without seed-specific tuning, the table is interpreted as a controlled transfer stress test.

The three refiner variants share the same Flow checkpoint and inference pipeline and differ in their diffusion-refiner weights. Flow proposals are generated from the checkpoint at training step $1{,}500$ using one Euler integration step. We apply the valid-action mask with indices $[0, 1, 2, 3, 4, 5, 6, 10, 11]$ and hard-clip the final Flow proposal in normalized action space before applying ADC. ADC uses the RDT scheduler's forward noising operation, `add_noise`, with a nominal ADC fraction of 0.2 and a configured diffusion timestep of 200. The scheduler has $1{,}000$ DDPM training timesteps, uses the `squaredcos_cap_v2` schedule, and predicts clean actions (`sample` prediction). One reverse-refinement step is followed by DCT-K2 postprocessing and conversion to ManiSkill actions, with an execution horizon of $H = 8$.

The Frozen variant uses the base RDT5000 checkpoint without further adaptation. Standard500 and Flow-conditioned Step50 both start from RDT5000 and update the same parameter subset: the final seven Transformer blocks (indices 21–27), the final action-output layer, and the diffusion-timestep embedding module. All remaining RDT parameters, the Flow model, condition projectors, vision and language encoders, and multimodal adapters remain fixed.

Standard500 uses the final checkpoint after 500 additional optimizer steps with the standard diffusion objective. Its noisy training inputs are sampled from the forward kernel centered on expert actions, and the refiner predicts the corresponding clean expert actions. At inference, this adapted

refiner uses the same ADC initialization process as the other hybrid variants. Standard500 thus tests the effect of further standard diffusion training.

Flow-conditioned adaptation applies ADC to proposals from the frozen Flow model and trains the refiner to predict the corresponding expert actions. Its objective combines the ADC-conditioned loss with a standard diffusion loss weighted by 0.25, preserving supervision on standard noisy expert-action inputs. The adaptation run lasts 500 optimizer steps, and Flow-conditioned Step50 uses the checkpoint at step 50, selected by validation performance. Both adapted variants use the same trainable parameter subset and evaluation pipeline.

Table 6: Matched ManiSkill closed-loop comparison. Environment-level progress metrics and per-chunk inference time are reported as mean ± standard deviation where available.

Method	Success	First grasp
Pure Diffusion	2/9 (22.2%)	2/9 (22.2%)
Pure Flow	3/9 (33.3%)	3/9 (33.3%)
Frozen	0/9 (0%)	0/9 (0%)
Standard500	2/9 (22.2%)	3/9 (33.3%)
Flow-conditioned Step50	**3/9 (33.3%)**	**3/9 (33.3%)**

Method	Min. TCP–cube dist. m ↓	Cube disp. m
Pure Diffusion	0.0437 ± 0.0702	0.0355 ± 0.0629
Pure Flow	0.0108 ± 0.0229	0.0488 ± 0.0726
Frozen	0	0
Standard500	0.0513 ± 0.0229	0.0375 ± 0.0726
Flow-conditioned Step50	$\mathbf{0.0108 \pm 0.0229}$	0.0488 ± 0.0726

Method	Action StepDiff P95 ↓	Chunk time s ↓
Pure Diffusion	0.0252 ± 0.0127	$\mathbf{0.1503 \pm 0.0281}$
Pure Flow	0.0208 ± 0.0075	0.2196 ± 0.0032
Frozen	0	0.2341 ± 0.0054
Standard500	0.0240 ± 0.0075	0.2339 ± 0.0050
Flow-conditioned Step50	$\mathbf{0.0208 \pm 0.0075}$	0.2329 ± 0.0024

The simulation results complement the offline evaluation of the generation mechanism. Flow-conditioned Step50 preserves the Pure Flow success rate ($3/9$) and approach statistics while retaining diffusion refinement. Relative to Standard500, it achieves one additional successful rollout ($3/9$ versus $2/9$), a lower mean minimum TCP–cube distance (0.0108 versus 0.0513 m), and lower Action StepDiff P95 (0.0208 versus 0.0240). Both adapted variants achieve first grasp in $3/9$ cases. These observations compare the selected refiner configurations; their training objectives, selected update counts, and checkpoint-selection procedures differ. Frozen achieves neither success nor first grasp ($0/9$ for both). Table 1 and Figure 2 report the controlled offline action results. Table 3 examines the effect of the ADC fraction.

B.6 External-policy transfer references

Table 7 reports environment-level metrics for three heterogeneous external policies evaluated under the same rollout grid. We report metrics with a consistent environment-level interpretation across policy families. We do not rank these policies by native-action smoothness or policy-call runtime. Their action formats, chunk interfaces, and policy-call definitions differ.

The dense metrics show why terminal success alone is insufficient for diagnosing transfer behavior: Diffusion Policy and OpenVLA reach the object in a substantial fraction of trials despite zero terminal success, whereas Octo shows little task progress under this interface. OpenVLA is deterministic in the evaluated action interface. Changing the sampling-seed index therefore does not alter its policy output. We retain all nine rollout settings to match the shared evaluation protocol.

Table 7: External-policy references on ManiSkill PickCube-v1 under the fixed 3×3 seed protocol. These results provide context for transfer difficulty and are not the main comparison.

Method	Success	First grasp
Diffusion Policy (Chi et al., 2023)	0/9 (0.0%)	**9/9 (100.0%)**
OpenVLA (Kim et al., 2025)	0/9 (0.0%)	6/9 (66.7%)
Octo (Ghosh et al., 2024)	0/9 (0.0%)	0/9 (0.0%)

Method	Min. TCP–cube dist. m ↓	Cube disp. m
Diffusion Policy (Chi et al., 2023)	**0.0030 ± 0.0014**	0.0176 ± 0.0082
OpenVLA (Kim et al., 2025)	0.0516 ± 0.0670	0.0870 ± 0.1022
Octo (Ghosh et al., 2024)	0.1678 ± 0.0093	0.000257 ± 0.000060