Hydra-DP3: Frequency-Aware Right-Sizing of 3D Diffusion Policies for Visuomotor Control

Jinhao Zhang[1*] **Zhexuan Zhou**[1*] **Huizhe Li**[1] **Yichen Lai**[1]
Wenlong Xia[1] **Haoming Song**[2] **Youmin Gong**[1] **Jie Mei**[1†]

[1]Harbin Institute of Technology, Shenzhen
[2]Shanghai Jiao Tong University

Abstract

Diffusion-based visuomotor policies perform well in robotic manipulation, yet current methods still inherit image-generation-style decoders and multi-step sampling. We revisit this design from a frequency-domain perspective. Robot action trajectories are highly smooth, with most energy concentrated in a few low-frequency discrete cosine transform modes. Under this structure, we show that the error of the optimal denoiser is bounded by the low-frequency subspace dimension and residual high-frequency energy, implying that denoising error saturates after very few reverse steps. This also suggests that action denoising requires a much simpler denoising model than image generation. Motivated by this insight, we propose Hydra-DP3 (**HDP3**), a pocket-scale 3D diffusion policy with a lightweight Diffusion Mixer decoder that supports two-step DDIM inference. Our synthetic experiments validate the theory and support the sufficiency of two-step denoising. Furthermore, across RoboTwin2.0, Adroit, MetaWorld, and real-world tasks, HDP3 achieves state-of-the-art performance with fewer than 1% of the parameters of prior 3D diffusion-based policies and substantially lower inference latency.

1 Introduction

Learning from demonstration provides a practical way to acquire visuomotor manipulation skills directly from expert data[1, 16], but standard behavior cloning often struggles with the multimodal nature of robotic actions, leading to mode averaging and compounding errors under distribution shift[24]. Diffusion-based policies address this limitation by modeling a conditional denoising process over action trajectories, and have shown strong performance in manipulation tasks[4]. Recent work has further extended this paradigm from 2D image observations to 3D point clouds, improving geometric robustness and transferability[39].

However, current 3D diffusion-based policies still largely inherit their design from image generation[39]. They typically combine a lightweight point-cloud encoder with a heavy decoder backbone, often a conditional U-Net[23] or DiT[17], and require multiple denoising steps at inference time. This is reasonable for reconstructing dense high-frequency image details, but robot action generation is fundamentally different: the target is a short, temporally smooth trajectory rather than a perceptual signal. This mismatch suggests that current 3D diffusion policies may be substantially overdesigned in both decoder capacity and denoising steps.

Our starting point is the frequency structure of robot trajectories. We find that action energy in frequency domain is highly concentrated in the lowest discrete cosine transform (DCT) modes, with the first two modes accounting for nearly all of the total energy. Building on this observation, we

*Equal contribution.
†Corresponding author: jmei@hit.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

provide a frequency-domain theoretical analysis showing that, when trajectory energy is concentrated in a shallow frequency-domain subspace, only a very small number of sampling steps is sufficient to drive the decoding error to a low level, thereby having negligible effect on the closed-loop execution of the trajectory. As a result, for smooth robot trajectories, performance should saturate after very few reverse steps (as observed in our experiments, two steps are already sufficient), and later denoising iterations offer diminishing returns.

Besides, recent theory on diffusion models shows that, when data are concentrated on a low-dimensional subspace, the complexity of score approximation is governed primarily by the intrinsic, rather than ambient, dimension of the data[2]. From this perspective, the strong concentration of robot actions in the frequency domain suggests that the score field to be learned for action denoising is substantially simpler than that of high-dimensional perceptual signals. Therefore, a large image-generation-style decoder may be unnecessary; what is needed instead is efficient fusion of temporal, channel-wise, and conditional information. Based on this principle, we propose Hydra-DP3 (**HDP3**), a pocket-scale 3D diffusion policy that replaces the heavy conditional decoder of prior methods with a lightweight **Diffusion Mixer (DiM)** decoder. With standard DDIM sampling[29], HDP3 supports two-step inference without consistency distillation or MeanFlow training. Across three simulation benchmarks—RoboTwin2.0, Adroit, and MetaWorld—it achieves state-of-the-art performance with fewer than 1% of the parameters of prior methods and substantially lower inference latency, while real-world experiments further validate its practicality beyond simulation. Our contributions are summarized as follows:

- We show that robot action trajectories are strongly low-frequency-dominant, revealing a mismatch in existing 3D diffusion policies, which still rely on heavy image-generation-style decoders and multi-step denoising.
- We provide a theoretical frequency-domain analysis showing that, for such trajectories, denoising error saturates quickly, so very few reverse steps are sufficient for effective action generation.
- We propose Hydra-DP3 (**HDP3**), a pocket-scale 3D diffusion policy with a lightweight **Diffusion Mixer (DiM)** decoder that supports efficient two-step DDIM inference.
- We validate the analysis and design on synthetic data, three simulation benchmarks, and real-world tasks, achieving state-of-the-art performance with fewer than 1% of the parameters of prior methods and substantially lower inference latency.

2 Related Work

2.1 Diffusion Models for Visuomotor Control.

Traditional Behavior Cloning (BC) often struggles with multimodal action distributions, leading to mode-averaging artifacts that result in suboptimal or unsafe behaviors in manipulation tasks. Diffusion Policy[4] addresses this fundamental limitation by formulating policy learning as a conditional denoising process over the action space. By learning the gradient of the action distribution, it effectively captures complex, multimodal human behaviors and provides superior stability in high-dimensional manipulation tasks compared to explicit regression policies. This probabilistic formulation has demonstrated remarkable expressiveness in capturing the inherent stochasticity of human demonstrations, outperforming prior generative approaches such as Implicit Behavior Cloning[7] and Conditional VAEs[28] across diverse benchmarks. Furthermore, the iterative refinement mechanism inherent to diffusion models enables the generation of temporally coherent action sequences, which is critical for contact-rich manipulation scenarios.

2.2 From 2D to 3D Representations.

While early diffusion policies relied predominantly on 2D RGB images as visual observations, such representations lack robustness against lighting variations, viewpoint changes, and domain shifts between simulation and real-world environments. Recent works leverage 3D point clouds processed by geometric encoders like PointNet++[19] to extract view-invariant spatial features that better capture object geometry and spatial relationships. 3D Diffusion Policy (DP3) [39] integrates these efficient 3D representations with diffusion models, achieving state-of-the-art data efficiency and

generalization capabilities in few-shot imitation learning settings. Complementary approaches such as Act3D[9] and PerAct[27] have further demonstrated the benefits of 3D-aware representations for language-conditioned manipulation. However, existing architectures typically pair lightweight point encoders with computationally heavy U-Net decoders for the diffusion backbone, creating a significant parameter redundancy and inference efficiency bottleneck that limits real-time deployment on resource-constrained robotic platforms.

2.3 Frequency-Aware Policies.

Recent works also exploit the frequency structure of robot actions. FreqPolicy-AR[41] represents trajectories through hierarchical DCT components and generates continuous tokens autoregressively from coarse to fine. FreqPolicy-Flow[31] introduces frequency-consistency constraints into flow-based training to support one-step action generation. H^3DP[14] aligns coarse-to-fine action generation with hierarchical visual features. Existing methods exploit frequency structure for action representations, additional training objectives, or hierarchical generation. In contrast, we analyze the complexity required for action denoising based on the spectral properties of action trajectories, thereby reducing both decoder capacity and the number of denoising steps.

2.4 Inference Acceleration.

To overcome the prohibitively high latency of iterative denoising during deployment, recent research has focused on principled approaches to sampling acceleration without sacrificing action quality. One-Step Diffusion Policy (OneDP)[34] employs knowledge distillation techniques to compress the multi-step diffusion process into a single forward pass, achieving real-time control frequencies of 62Hz suitable for reactive manipulation. Consistency Policy[18] takes an alternative approach by enforcing self-consistency constraints along probability flow ordinary differential equation (ODE) trajectories, enabling high-quality action generation in just 1-2 denoising steps while preserving the multimodal expressiveness of the original diffusion formulation. Drawing inspiration from recent advances in generative modeling, FlowPolicy[40] utilizes Consistency Flow Matching[37] to learn straight-line ODE paths between noise and data distributions, enabling faster, more numerically stable inference. MP1[26] employs MeanFlow[8] for one-step generation without timestep conditioning. These acceleration techniques collectively represent a promising direction toward bridging the gap between the representational power of diffusion-based policies and the stringent latency requirements of closed-loop robotic control.

3 Problem Setup and Policy Overview

We learn an end-to-end visuomotor policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$ from expert demonstrations of observation–action pairs, where each observation consists of a single-view, robot-centric point cloud $\mathbf{P}$ and robot proprioceptive state $\mathbf{s}$. Following the DDPM formulation[10], we model action generation as a conditional denoising process. Given a clean action trajectory x_0, the forward process corrupts it at timestep t as

$$x_t = \alpha_t x_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where α_t and σ_t are determined by the noise schedule. Our model adopts the x_0-prediction parameterization[22] and directly predicts the clean trajectory $\hat{x}_0$ from the noisy input x_t.

As shown in Fig. 2, our framework consists of a compact observation encoder and a lightweight diffusion decoder. Following DP3[39], we first downsample $\mathbf{P}$ using farthest point sampling and encode the sampled points with an MLP followed by max pooling to obtain a compact 3D feature $\mathbf{z}$. We then linearly project the robot state and add it to $\mathbf{z}$ to form the conditioning context:

$$C = \mathbf{z} + \mathrm{Linear}(\mathbf{s}). \tag{2}$$

Conditioned on C and the diffusion timestep t, the decoder takes the noisy trajectory x_t as input, projects it into a sequence of tokens, processes them through K DiM blocks, and maps the fused features to the final prediction $\hat{x}_0$. Details of the DiM block are provided in Sec. 4.3.

During training, we optimize the reconstruction objective

$$\mathcal{L} = \mathbb{E}_{(x_0, \mathbf{P}, \mathbf{s}), t, \epsilon} \left[\|\hat{x}_0 - x_0\|_2^2 \right]. \tag{3}$$

At inference time, we use DDIM sampling[29] to accelerate denoising. For highly smooth trajectory data, we find that, in practice, two denoising steps are sufficient without requiring additional consistency distillation techniques[18, 35, 40].

4 Method

4.1 Motivation

Most existing diffusion-based visuomotor policies[4, 39] rely on multi-step DDPM[10] or DDIM[29] denoising schedules together with large U-Net[23] or DiT[17] decoders. However, prior work has largely overlooked a key property of robot trajectory data: these trajectories are highly smooth in the time domain, and thus their energy is concentrated primarily in low-frequency bands in the frequency domain as shown in [32]. Exploiting this property, we show both theoretically and empirically that, for robot trajectory data, just **two DDIM denoising steps** are sufficient for the decoding accuracy (measured by MSE) and rollout success rate to nearly saturate, **without** requiring additional sampling acceleration techniques such as consistency distillation[30] or MeanFlow[8]. Furthermore, motivated by the fact that action data inherently lie on a low-dimensional manifold, and by recent theory showing that score approximation in diffusion models depends on the intrinsic rather than ambient dimension of the data[2], we propose a much lighter decoder architecture that uses less than 1% of the parameters of prior methods while surpassing previous state-of-the-art performance.

To further quantify the low-frequency-dominant nature of robot trajectories, we apply the discrete cosine transform (DCT) to map them into the frequency domain and measure the energy proportion in each frequency band. Specifically, we collect 2,500 episodes across all 50 tasks in RoboTwin2.0 and segment them into approximately 20,000 trajectories with a sequence length of $T = 32$. We then apply the DCT and compute the energy ratio of each DCT mode, as shown in Figure 1. As can be seen, the first two DCT modes account for 98.5% of the total energy, with the vast majority concentrated in the $k = 1$ mode alone (93.2%). This indicates that robot trajectory data lies on a low-dimensional manifold in the frequency domain and is highly concentrated in the low-frequency regime. More details on the frequency-domain decomposition and the energy distribution for different tasks are provided in Appendix D.

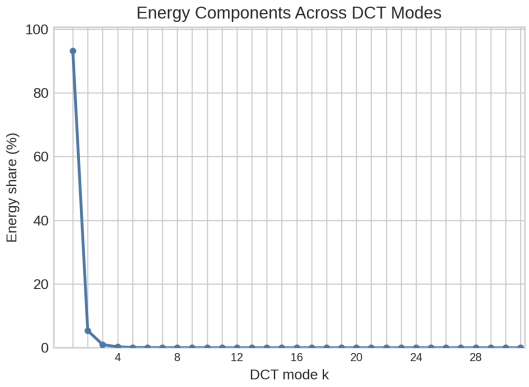


Figure 1: Frequency structure of action trajectories.

4.2 Two-Step Denoising Suffices for Robot Action Data

Prior work has provided both theoretical and empirical evidence that diffusion models tend to first generate low-frequency structure during sampling and then refine high-frequency details[6]. Intuitively, for robot trajectories, where low-frequency components are overwhelmingly dominant, we expect the low-frequency portion to be reconstructed quickly in the early sampling steps. Subsequent refinement of the high-frequency components, which are mainly attributed to sensor noise and other minor perturbations[32], becomes unnecessary. As a result, only a small number of decoding steps should be sufficient to reach saturation performance.

To formalize this intuition, we study the denoising error in the frequency domain. Let x_0 denote the original data, and let y_0 denote its frequency-domain representation obtained through an orthogonal DCT. For analytical convenience, we assume that $y_0 = \mathbf{U}x_0$ is distributed as a zero-mean Gaussian with covariance $\mathbf{\Sigma}$. Let $\mathbf{P}_L = \mathrm{diag}(1, \ldots, 1, 0, \ldots, 0)$ be a diagonal matrix with the first m diagonal entries equal to 1 and the remaining entries equal to 0, and let $\mathbf{P}_H = \mathbf{I} - \mathbf{P}_L$. The low-frequency-dominant nature of trajectory data can then be expressed as follows:

$$\mathrm{Tr}(\mathbf{P}_H \mathbf{\Sigma}) \leq \eta \, \mathrm{Tr}(\mathbf{\Sigma}) \tag{4}$$

where $\mathrm{Tr}(\cdot)$ denotes the matrix trace, and η denotes the fraction of energy contained in the high-frequency components. Consider the forward noising process $x_t = \alpha_t x_0 + \sigma_t \epsilon$. Then, as shown in [5], the optimal denoiser is given by the posterior expectation $x_0^* = \mathbb{E}[x_0|x_t]$. Because $\mathbf{U}$ is orthogonal, the noising process and the optimal estimator admit the same form in the frequency domain:

$$y_t = \alpha_t y_0 + \sigma_t \xi, \quad \xi \sim \mathcal{N}(0, \mathbf{I}) \tag{5}$$

$$y_0^* = \mathbf{U}x_0^* = \mathbf{U}\mathbb{E}[x_0|x_t] = \mathbb{E}[\mathbf{U}x_0|\mathbf{U}x_t] = \mathbb{E}[y_0|y_t] \tag{6}$$

The error of this optimal estimator in the frequency domain satisfies the following theorem.

Theorem 4.1 (Frequency-Domain Error Bound of the Optimal Estimator)**.** *Under the above assumptions, the optimal estimation errors in the low- and high-frequency components satisfy, respectively:*

$$e_L = \frac{1}{n}\mathbb{E}\|\mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} \tag{7}$$

$$e_H = \frac{1}{n}\mathbb{E}\|\mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \leq \frac{\eta}{n}\mathrm{Tr}(\mathbf{\Sigma}) \tag{8}$$

where n is the number of frames in the predicted trajectory, e_L and e_H denote the mean squared errors (MSEs) of the low- and high-frequency components, respectively. Consequently, the total error satisfies:

$$e = \frac{1}{n}\mathbb{E}\|x_0 - \mathbb{E}[x_0|x_t]\|^2 = \frac{1}{n}\mathbb{E}\|y_0 - \mathbb{E}[y_0|y_t]\|^2 = e_L + e_H \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} + \frac{\eta}{n}\mathrm{Tr}(\mathbf{\Sigma}) \tag{9}$$

Normalizing by the average per-frame energy, the relative error is bounded by:

$$\hat{e} = \frac{e}{\mathrm{Tr}(\mathbf{\Sigma})/n} \leq \frac{m}{\mathrm{Tr}(\mathbf{\Sigma})}\frac{\sigma_t^2}{\alpha_t^2} + \eta \tag{10}$$

Proof. See App. A. □

Remark 4.2. We can further relax the assumptions of the theorem to cover non-Gaussian distributions. See App. A.2 for the proof.

Remark 4.3. The above theorem shows that **the more concentrated the energy of the data is in the DCT spectrum, i.e., the smaller m and η, the smaller the average per-frame error of the optimal estimator**.

For a T-step DDIM sampler, its final output can be interpreted as a one-step estimate of x_0 starting from $t = \frac{1}{T}$. This yields the following corollary for the MSE of T-step sampling.

Corollary 4.4. *The upper bound on the relative error $\hat{e}_T$ of a T-step DDIM sampler can be estimated as follows:*

$$\hat{e}_T \lesssim \frac{m}{\mathrm{Tr}(\mathbf{\Sigma})}\frac{\sigma_{T-1}^2}{\alpha_{T-1}^2} + \eta \tag{11}$$

Consistent with the preceding analysis, this indicates that greater spectral concentration of the data leads to lower DDIM decoding error for the same number of denoising steps. In our experiments, we find that two denoising steps are already sufficient for robot trajectories. In addition, we find that, for low-frequency-dominant data, the error under few-step decoding mainly comes from the high-frequency band, and this component can be further suppressed by the low-level controller, making the resulting closed-loop performance nearly indistinguishable from that of multi-step decoding (See App. B for details). The synthetic data experiments in Sec. 5 and the rollout evaluations in Sec. 6.3 clearly support that two-step decoding is sufficient in most cases.

Remark 4.5. We can also explain the effectiveness of few-step sampling from the perspective of reduced crossings among interpolation trajectories: the error in Eq. 9 equals the expected posterior variance per frame, $\frac{1}{n}\mathbb{E}_{x_t}[\mathrm{Tr}\,\mathrm{Cov}(x_0|x_t)]$, and a smaller value indicates a more concentrated and stable prediction target. This provides a posterior-uncertainty motivation for few-step sampling[36, 11].

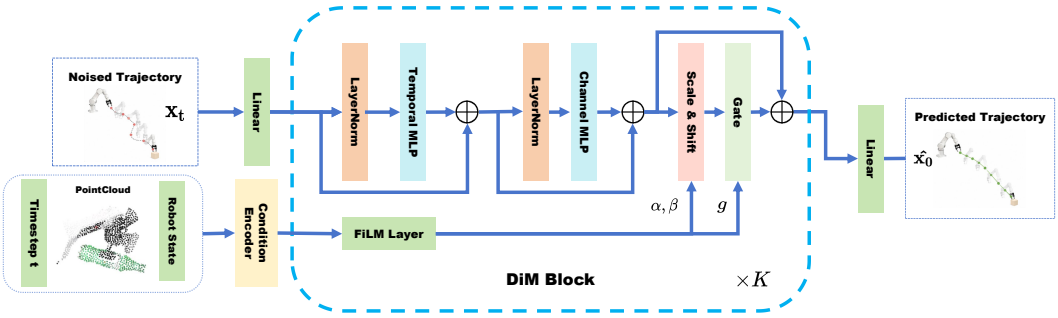


Figure 2: **Overall architecture of the Proposed Method.** In the figure, T denotes transpose. We adopt the efficient point-cloud encoder from DP3[39] and stacks K DiM blocks as the decoder. Each DiM block is built upon an MLP-Mixer style[33] architecture, enabling efficient information fusion with a small parameter budget, thereby improving decision-making performance.

4.3 Lightweight Decoder for Minimal Denoising

Recent theory on diffusion models shows that, when data are concentrated on a low-dimensional subspace, the complexity of score approximation is governed primarily by the intrinsic, rather than ambient, dimension of the data[2]. Combined with the strong low-frequency concentration of robot action trajectories observed above, this suggests that the score field to be learned for action denoising is substantially simpler than that of high-dimensional perceptual signals. Therefore, instead of relying on a heavy high-capacity decoder, we seek a lightweight architecture that is better matched to the intrinsic complexity of robot action data. Specifically, we adopt MLP-Mixer[33] as a simple yet effective decoder backbone. Using only lightweight transpose operations and MLP-based projections, MLP-Mixer enables efficient interaction across both temporal and channel dimensions while maintaining stable training. As shown in Fig. 2, given the token sequence H obtained by projecting the noisy trajectory, we first transpose it and feed it into a temporal MLP to aggregate information along the time dimension. We then transpose it back and add a residual connection. Next, we apply the same procedure along the channel dimension, yielding a representation that integrates both temporal and channel-wise information:

$$H = H + \mathrm{MLP}(H^{\mathrm{T}})^{\mathrm{T}}; \quad H = H + \mathrm{MLP}(H) \tag{12}$$

Finally, we inject the conditioning information into the model using a FiLM layer combined with a gated residual connection:

$$\alpha, \beta, g = \mathrm{MLP}(H, C) \tag{13}$$

$$\tilde{H} = H \odot \alpha + \beta \tag{14}$$

$$H = \tilde{H} \odot g + H \tag{15}$$

After stacking K such blocks, we obtain the final tokens in which the conditioning information is fully mixed, which are then used to predict the denoised actions:

$$\hat{x}_0 = \mathrm{Linear}(H) \tag{16}$$

Unlike U-Net and DiT, which rely on large intermediate dimensions to improve denoising capacity, our design substantially enhances the efficiency of information fusion, allowing us to surpass previous state-of-the-art performance using **less than 1%** of the parameters of prior methods.

5 Synthetic Data Experiments

To validate the theory in Sec. 4.2, we design a corresponding synthetic-data experiment. Specifically, we synthesize three classes of trajectories: a low-frequency-dominant (`lowfreq`) dataset, where the first 6 out of 64 DCT modes account for 99% of the total energy; a `broadband` dataset, where energy is evenly distributed across all 64 modes; and a high-frequency-dominant (`highfreq`) dataset, where the last 6 out of 64 DCT modes account for 99% of the total energy. We normalize all datasets to a common scale to ensure a fair comparison in training and evaluation. We then train identical

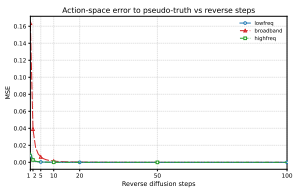


Figure 3: MSE Under Different NFEs

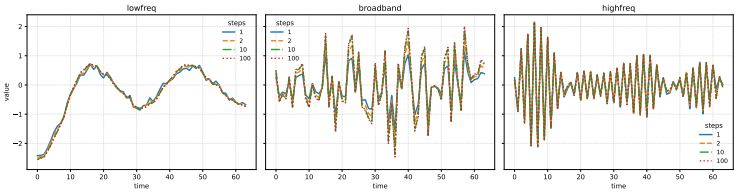


Figure 4: Examples of Decoded Trajectories

Table 1: Evaluation on the Robotwin2.0 benchmark. Each task is tested across 100 randomly generated scenes using 100 different seeds.

Method	Open Microwave	Hanging Mug	Move Pillbottle Pad	Stamp Seal	Beat Block Hammer	Place Phone Stand	...	Average (50 tasks)
DP	5.0	8.0	1.0	2.0	42.0	13.0	...	28.0
DP3	61.0	17.0	41.0	18.0	72.0	44.0	...	55.2
Flow Policy	17.0	6.0	14.0	20.0	75.0	44.0	...	41.1
MP1	35.0	11.0	35.0	28.0	68.0	52.0	...	55.2
HDP3 **(Ours)**	**92.0** (+31.0)	**35.0** (+18.0)	**56.0** (+15.0)	**43.0** (+15.0)	**88.0** (+13.0)	**65.0** (+13.0)	...	**63.2** (+8.0)

CNN-based diffusion models on these data and evaluate the MSE under different numbers of decoding (reverse) steps, taking the 100-step decoding result as the reference. Detailed descriptions of data synthesis, training and evaluation protocols, and additional results are provided in App. B.

The results are shown in Fig. 3, Fig. 4. As can be seen, for the same number of decoding steps, the `lowfreq`/`highfreq` trajectories achieve substantially lower denoising error than the `broadband` trajectories. Moreover, the error on `lowfreq`/`highfreq` nearly saturates after just two reverse steps, whereas `broadband` requires many more decoding steps to reach comparable performance. We further observe that, under few-step decoding, most of the remaining error for `lowfreq` lies in the high-frequency band. In practice, the robot's low-level controller further filters out this high-frequency error, making two-step decoding nearly indistinguishable from using more reverse steps. See App. B for details.

6 Simulation Experiments

6.1 Experimental Setup

Simulation Benchmarks. We evaluate our method on three widely used benchmarks: RoboTwin2.0[3], Adroit[21], and MetaWorld[38]. RoboTwin2.0 focuses on dual-arm manipulation with diverse assets and scripted data generation. Adroit targets high-dimensional dexterous control for precise, long-horizon skills, while MetaWorld offers tiered single-arm tasks[25]. Collectively, these benchmarks cover both single- and dual-arm settings across varying dimensionalities, ensuring a comprehensive evaluation.

Training and Evaluation Details. For a fair comparison, we follow the official training and evaluation protocols for all benchmarks.

- For RoboTwin2.0, we train the models using 50 expert demonstrations. We evaluate each task on 100 randomly generated scenes and report the success rate over these 100 evaluation episodes.
- For Adroit and MetaWorld, we use 10 expert demonstrations to assess the data efficiency of the methods. We run three independent trials with random seeds $0, 1, 2$. During training, we evaluate the policy every 200 epochs; each evaluation consists of 20 rollouts per task, from which we compute the success rate. For each seed, we track the success rate over training and define SR_5 as the average of the top five success rates; we report the average SR_5 across the three seeds for all methods.

We adopt a DDIM[29] noise scheduler with 100 diffusion steps during training and 2 steps at inference, and optimize using AdamW[13] with an initial learning rate of 1×10^{-4} and a cosine decay schedule.

Table 2: Evaluation on the Adroit and MetaWorld benchmark. Each task is trained and tested across 3 different seeds. Unavailable results are denoted by "–".

Method	Hammer	Door	Pen	Assembly	Disassemble	Hand-Insert	Pick-Place-Wall	Push	Reach-Wall	Stick-Push	Avg.
BCRNN	0.0	0.0	9.0	3.0	32.0	–	–	–	–	–	8.8
IBC	0.0	0.0	9.0	0.0	1.0	–	–	–	–	–	2.0
DP	45.0	37.0	13.0	15.0	43.0	9.0	5.0	11.0	59.0	63.0	30.0
DP3	**100.0**	62.0	43.7	99.6	75.0	25.3	82.7	71.3	70.7	**100.0**	73.0
Flow Policy	**100.0**	**65.3**	48.2	92.7	60.7	18.6	88.3	79.7	72.7	**100.0**	72.6
MP1	**100.0**	56.5	44.0	**100.0**	**93.7**	19.3	**91.3**	79.3	82.7	**100.0**	76.7
HDP3 (**Ours**)	**100.0**	57.3	**48.7**	**100.0**	87.7	**33.3**	88.7	**83.0**	**85.0**	**100.0**	**78.4**

The hidden dimension and depth of the DiM block are set to 128 and 6, respectively. To stabilize training, both actions and robot states are normalized to $[-1, 1]$. All models are trained for 3,000 epochs on a single NVIDIA RTX 5880 Ada GPU, using a batch size of 256 for RoboTwin 2.0 and 128 for Adroit and MetaWorld. Detailed hyperparameter settings are provided in Appendix C.

Baselines. On RoboTwin2.0, we compare our method with the sota diffusion-based approaches, namely DP3[39], DP[4], Flow Policy[40] and MP1[26]. On Adroit and MetaWorld, we include all the aforementioned diffusion-based methods, and further compare against two commonly used imitation-learning baselines, IBC[7] and BCRNN[15].

6.2 Comparison with State-of-the-art Methods

Model Size and Inference Latency. We measure the model size and inference latency of several diffusion-based policies; the results are reported in Tab. 3. Thanks to two-step inference and a compact model size, our HDP3 substantially reduces inference latency compared to the baseline method, DP3, while maintaining strong performance (as shown below). Moreover, while Flow Policy and MP1 achieve impressive latency via 1-step inference, it retains a massive parameter count (∼256M). HDP3 achieves comparable low latency (∼4.50ms) but with **two orders of magnitude fewer parameters**, significantly lowering the memory footprint for on-device deployment.

Table 3: **Model size and inference latency comparison.** All evaluations are conducted on a single RTX 5880 Ada GPU with the batch size set to 1.

Method	Params (M)↓	NFE	Latency (ms)↓
DP	85.6	100	460
DP3	255.8	10	51.4
Flow Policy	255.8	**1**	7.04
MP1	255.8	**1**	7.02
HDP3	**2.52**	2	**4.50**

RoboTwin 2.0. As shown in Tab. 1, HDP3 achieves the best overall performance on RoboTwin2.0, reaching an average success rate of **63.2%** over 50 tasks. With only **two inference steps**, HDP3 outperforms the previous best baseline, DP3 (55.2%) and MP1 (55.2%), by **8.0 percentage points**, while also surpassing Flow Policy (40.9%). The six tasks shown in Tab. 1 correspond to the largest margins over the runner-up method: *Open Microwave*(+31.0), *Hanging Mug*(+18.0), *Move Pillbottle Pad* (+15.0), *Stamp Seal*(+15.0), *Beat Block Hammer*(+13.0), and *Place Phone Stand*(+13.0). These results highlight the consistent advantage of HDP3 across diverse manipulation tasks.

Adroit and MetaWorld. Tab. 2 compares HDP3 with prior methods across 10 manipulation tasks on Adroit and MetaWorld. HDP3 achieves the best overall performance with **78.4%** average success, outperforming the strongest baseline MP1 (76.7%) by **1.7 percentage points**, and exceeding DP3 (73.0%) and Flow Policy (72.6%) by 5.4 and 5.8 percentage points, respectively. HDP3 attains the best performance on *Pen*, *Hand-Insert*, *Push*, and *Reach-Wall*, while matching the best result on *Hammer*, *Assembly*, and *Stick-Push*. Despite using an extremely small number of parameters and only two inference steps, HDP3 maintains—and in some cases surpasses—the performance of prior state-of-the-art methods.

6.3 Ablation Studies

Necessity of the Mixer-based Architecture. To validate the design of HDP3, we conduct an ablation study that compares different architectures under the same parameter budget. The results are summarized in Tab. 4. Here, Vanilla-MLP denotes a pure MLP architecture with FiLM-based conditioning, while Vanilla-UNet is obtained by reducing the intermediate channel width in the

Table 4: **Ablation on design choices in HDP3.** To ensure a fair comparison, we tune the hidden size for each setting to keep the number of parameters comparable.

Method	Num.Params.(M)	Door		Hammer		Pen		Avg.	
		SR_5	Loss	SR_5	Loss	SR_5	Loss	SR_5	Loss
HDP3	2.52	**57.3**	6e-4	**100**	2e-4	**48.7**	5e-4	**68.7**	4.3e-4
Vanilla-MLP	2.52	0.0	3e-2	0.0	1e-2	0.0	3e-2	0.0	2.3e-2
Vanilla-UNet	2.64	25.7	3e-3	93.3	3e-4	42.0	1e-3	53.7	1.4e-3

Table 5: **Component ablations of DiM.** Each variant changes one component of the proposed DiM block.

Configuration	Door	Hammer	Pen
HDP3	**57.3**	**100.0**	**48.7**
w/o Temporal Mixing	0.0	0.0	9.0
w/o Channel Mixing	45.5	98.3	39.3
w/o Normalization	48.8	**100.0**	42.5
w/o Gate	55.8	98.7	43.5

decoder of the original U-Net. We observe that the pure MLP baseline achieves an average SR_5 of 0 across all three tasks. Specifically, the training loss for Vanilla-MLP converged extremely poorly, indicating that a plain MLP is far from sufficient for robotic manipulation. Notably, our DiM block—which augments the MLP with residual connections and a temporal fusion module (incurring almost no additional parameters)—substantially improves performance. Furthermore, the U-Net variant with reduced channel width suffers a significant performance drop compared to the original DP3 and is substantially outperformed by our HDP3. These results suggest that, under a strict parameter budget, the DiM block offers superior parameter efficiency and information fusion capabilities compared to standard U-Net architectures.

Ablation of Individual Components. We further investigate the role of each component in the DiM block, including temporal mixing, channel mixing, normalization, and gating. The results are shown in Tab. 5. Removing temporal mixing nearly collapses performance, reducing the success rates on Door and Hammer to zero. This is consistent with the failure of Vanilla-MLP and indicates that temporal interaction is crucial for action denoising. Removing channel mixing, normalization, or gating also reduces performance. Therefore, temporal mixing is the most critical component, highlighting the importance of information exchange and fusion across timesteps.

Impact of the Number of Function Evaluations (NFE). We further demonstrate that two-step decoding is sufficient for robot trajectory generation through rollout success-rate evaluations on three Adroit tasks: *Door*, *Hammer*, and *Pen*. Specifically, we evaluate the final checkpoint after 3000 training epochs. For each NFE setting, we run 5,000 evaluation episodes to reduce statistical variance and report the mean score. The results are shown in Tab. 6. As can be seen, using only one inference step leads to a noticeable performance drop compared to multi-step settings. However, two-step inference already achieves strong performance, and additional steps (e.g., 5 or 10) do not yield further gains. This indicates that multi-step decoding is not necessary, and that two-step decoding is already sufficient for closed-loop execution.

Table 6: **Ablation on NFE.** We report the average success rate (%) over 5,000 randomly generated scenes for each task using the final checkpoint (3,000 epochs). *NFE* denotes the Number of Function Evaluations.

	HDP3				DP3			
NFE	Door	Hammer	Pen	Avg.	Door	Hammer	Pen	Avg.
1	34.2	91.5	33.4	53.0	44.6	100.0	27.8	57.5
2	37.4	100.0	37.2	58.2	47.4	100.0	30.5	59.3
5	37.7	100.0	35.5	57.7	46.1	100.0	31.7	59.3
10	36.8	100.0	37.3	58.0	47.8	100.0	31.8	59.9

6.4 Real-world Experiments

Experiment Settings. We validate the effectiveness of our method on an AgileX Piper robot. Real-world visual observations are captured with a single Intel RealSense D455 camera mounted globally. The model is deployed on an NVIDIA RTX 4060 GPU for on-board action inference. We collect expert demonstrations via teleoperation using a leader-follower setup, yielding 50 trajectories for training. The action space is defined as 6-DoF joint positions. See App. C for more implementation details.

Results. We report the success rates across 45 real-world trials per task and method, organized into three sets of 15, in Tab. 7. We report the mean and standard deviation across the three sets. Consistent with our simulation findings, HDP3 maintains strong performance in these real-world experiments. It outperforms the DP3 baseline by 16.3 percentage points on average and also outperforms the DP baseline on all three tasks.

Table 7: **Real-world Experiments Success Rate (%).**

Task	DP	DP3	HDP3 **(Ours)**
Place Object	33.3 ± 10.9	48.9 ± 3.2	$\mathbf{75.5} \pm 3.2$ (↑**26.6**)
Adjust Bottle	35.5 ± 3.2	37.8 ± 3.2	$\mathbf{46.7} \pm 5.5$ (↑**8.9**)
Stack Blocks Two	8.9 ± 3.2	6.7 ± 5.5	$\mathbf{20.0} \pm 5.5$ (↑**13.3**)

7 Conclusion

In this work, we introduced **HDP3**, a compact yet powerful 3D visuomotor policy. Motivated by the insight that robot trajectories are strongly dominated by low-frequency components in the frequency domain, we provide a theoretical analysis showing that two-step decoding does not affect closed-loop execution, and further replace the parameter-heavy conditional decoder with a lightweight MLP-Mixer-based DiM block. Synthetic-data experiments validate the theory-driven intuition, while simulation results show that HDP3 achieves state-of-the-art performance with less than 1% of the parameters of prior methods. Real-world experiments further demonstrate the practical applicability of the proposed method. For limitations, please refer to App. H.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. In *International Conference on Machine Learning*, pages 4672–4712. PMLR, 2023.

[3] 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.

[4] 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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[5] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

[6] Fabian Falck, Teodora Pandeva, Kiarash Zahirnia, Rachel Lawrence, Richard Turner, Edward Meeds, Javier Zazo, and Sushrut Karmalkar. A fourier space perspective on diffusion models. *arXiv preprint arXiv:2505.11278*, 2025.

[7] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[8] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[9] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *Conference on Robot Learning*, pages 3949–3965. PMLR, 2023.

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

[11] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[12] Zhenyang Liu, Yikai Wang, Kuanning Wang, Longfei Liang, Xiangyang Xue, and Yanwei Fu. Spatial-temporal aware visuomotor diffusion policy learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–10. IEEE, 2025.

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

[14] Yiyang Lu, Yufeng Tian, Zhecheng Yuan, Xianbang Wang, Pu Hua, Zhengrong Xue, and Huazhe Xu. H∋3 dp: Triply-hierarchical diffusion policy for visuomotor learning. In *International Conference on Learning Representations*, volume 2026, pages 13237–13261, 2026.

[15] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

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

[17] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[18] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[20] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

[21] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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

[25] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[26] 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, pages 18532–18539, 2026.

[27] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[28] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[30] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

[31] Yifei Su, Ning Liu, Dong Chen, Zhen Zhao, Kun Wu, Meng Li, Zhiyuan Xu, Zhengping Che, and Jian Tang. Freqpolicy: Efficient flow-based visuomotor policy via frequency consistency. *Advances in Neural Information Processing Systems*, 38:27769–27797, 2026.

[32] Hengkai Tan, Songming Liu, Kai Ma, Chengyang Ying, Xingxing Zhang, Hang Su, and Jun Zhu. Fourier controller networks for real-time decision-making in embodied learning. *arXiv preprint arXiv:2405.19885*, 2024.

[33] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[34] Zhendong Wang, Max 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 *Forty-second International Conference on Machine Learning*, 2025.

[35] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[36] Donglin Yang, Yongxing Zhang, Xin Yu, Liang Hou, Xin Tao, Pengfei Wan, Xiaojuan Qi, and Renjie Liao. Stable velocity: A variance perspective on flow matching. *arXiv preprint arXiv:2602.05435*, 2026.

[37] Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[38] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[39] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *2nd Workshop on Dexterous Manipulation: Design, Perception and Control (RSS)*, 2024.

[40] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 14754–14762, 2025.

[41] Yiming Zhong, Yumeng Liu, Chuyang Xiao, Zemin Yang, Youzhuo Wang, Yufei Zhu, Ye Shi, Yujing Sun, Xinge Zhu, and Yuexin Ma. Freqpolicy: Frequency autoregressive visuomotor policy with continuous tokens. *Advances in Neural Information Processing Systems*, 38:56493–56526, 2026.

A Proof and Extensions of the Frequency-Domain Analysis

A.1 Proof of Theorem 4.1

We first derive the closed-form expression of the optimal estimator in the frequency domain, given in the following lemma.

Lemma A.1. *Suppose that $y_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ and $y_t = \alpha_t y_0 + \sigma_t \xi$, where $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, the posterior distribution is Gaussian, with mean and covariance given by:*

$$\mathbb{E}[y_0|y_t] = \alpha_t \mathbf{\Sigma}(\alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I})^{-1} y_t \tag{17}$$

$$\mathrm{Cov}[y_0|y_t] = \sigma_t^2 \mathbf{\Sigma}(\alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I})^{-1} \tag{18}$$

Proof. Since y_0 and ξ are independent, the pair (y_0, ξ) is jointly Gaussian. Moreover, since we have:

$$\begin{bmatrix} y_0 \\ y_t \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \alpha_t \mathbf{I} & \sigma_t \mathbf{I} \end{bmatrix} \begin{bmatrix} y_0 \\ \xi \end{bmatrix} \tag{19}$$

Therefore, (y_0, y_t) is also jointly Gaussian and satisfies:

$$(y_0, y_t) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}) \tag{20}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{\Sigma} & \alpha_t \mathbf{\Sigma} \\ \alpha_t \mathbf{\Sigma} & \alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I} \end{bmatrix} \tag{21}$$

Hence, the posterior distribution is still Gaussian, and its mean and covariance are given by:

$$\mathbb{E}[y_0|y_t] = \mathrm{Cov}(y_0, y_t)\,\mathrm{Cov}(y_t, y_t)^{-1} y_t = \alpha_t \mathbf{\Sigma}(\alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I})^{-1} y_t \tag{22}$$

Meanwhile,

$$\mathrm{Cov}[y_0|y_t] = \mathrm{Cov}(y_0, y_0) - \mathrm{Cov}(y_0, y_t)\,\mathrm{Cov}(y_t, y_t)^{-1}\,\mathrm{Cov}(y_t, y_0) \tag{23}$$

By setting $\mathbf{A} = \alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I}$, we obtain:

$$\mathrm{Cov}[y_0|y_t] = \mathbf{\Sigma} - \alpha_t^2 \mathbf{\Sigma} \mathbf{A}^{-1} \mathbf{\Sigma} \tag{24}$$

$$= \mathbf{\Sigma} \mathbf{A}^{-1} (\mathbf{A} - \alpha_t^2 \mathbf{\Sigma}) \tag{25}$$

$$= \sigma_t^2 \mathbf{\Sigma} \mathbf{A}^{-1} \tag{26}$$

$$= \sigma_t^2 \mathbf{\Sigma}(\alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I})^{-1} \tag{27}$$

□

We next derive bounds on the posterior covariance, as stated in the following lemma.

Lemma A.2. *Under the Loewner partial order, the posterior covariance satisfies:*

$$\mathbf{0} \preceq \mathrm{Cov}[y_0|y_t] \preceq \mathbf{\Sigma} \tag{28}$$

$$\mathbf{0} \preceq \mathrm{Cov}[y_0|y_t] \preceq \frac{\sigma_t^2}{\alpha_t^2} \mathbf{I} \tag{29}$$

Proof. Let the spectral decomposition of $\mathbf{\Sigma}$ be

$$\mathbf{\Sigma} = \mathbf{Q}\,\mathrm{diag}(\nu_1, \ldots, \nu_n)\mathbf{Q}^\top \tag{30}$$

where $\mathbf{Q}$ is an orthogonal matrix, and ν_i denotes the nonnegative eigenvalues. Setting $\mathbf{A} = \alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I}$, we obtain:

$$\mathbf{A} = \alpha_t^2 \mathbf{\Sigma} + \sigma_t^2 \mathbf{I} \tag{31}$$

$$= \mathbf{Q}\,\mathrm{diag}(\alpha_t^2 \nu_1, \ldots, \alpha_t^2 \nu_n)\mathbf{Q}^\top + \mathbf{Q}\,\mathrm{diag}(\sigma_t^2, \ldots, \sigma_t^2)\mathbf{Q}^\top \tag{32}$$

$$= \mathbf{Q}\,\mathrm{diag}(\alpha_t^2 \nu_1 + \sigma_t^2, \ldots, \alpha_t^2 \nu_n + \sigma_t^2)\mathbf{Q}^\top \tag{33}$$

Thus,

$$\mathbf{A}^{-1} = \mathbf{Q}\,\mathrm{diag}\left(\frac{1}{\alpha_t^2 \nu_1 + \sigma_t^2}, \ldots, \frac{1}{\alpha_t^2 \nu_1 + \sigma_t^2}\right)\mathbf{Q}^\top \tag{34}$$

Furthermore, we have:

$$\mathrm{Cov}[y_0|y_t] = \sigma_t^2 \mathbf{\Sigma} \mathbf{A}^{-1} \tag{35}$$

$$= \mathbf{Q}\,\mathrm{diag}(\sigma_t^2\nu_1, \ldots, \sigma_t^2\nu_n)\mathbf{Q}^\top \mathbf{A}^{-1} \tag{36}$$

$$= \mathbf{Q}\,\mathrm{diag}(\frac{\sigma_t^2\nu_1}{\alpha_t^2\nu_1 + \sigma_t^2}, \ldots, \frac{\sigma_t^2\nu_n}{\alpha_t^2\nu_n + \sigma_t^2})\mathbf{Q}^\top \tag{37}$$

Consider the scalar function $f(\lambda) = \frac{\sigma_t^2\lambda}{\alpha_t^2\lambda + \sigma_t^2}$. Clearly, for $\lambda \geq 0$, we have:

$$0 \leq f(\lambda) \leq \lambda \tag{38}$$

$$0 \leq f(\lambda) \leq \frac{\sigma_t^2}{\alpha_t^2} \tag{39}$$

Thus, we obtain:

$$\mathbf{0} \preceq \mathrm{Cov}[y_0|y_t] \preceq \mathbf{Q}\,\mathrm{diag}(\nu_1, \ldots, \nu_n)\mathbf{Q}^\top = \mathbf{\Sigma} \tag{40}$$

$$\mathbf{0} \preceq \mathrm{Cov}[y_0|y_t] \preceq \mathbf{Q}\,\mathrm{diag}(\frac{\sigma_t^2}{\alpha_t^2}, \ldots, \frac{\sigma_t^2}{\alpha_t^2})\mathbf{Q}^\top = \frac{\sigma_t^2}{\alpha_t^2}\mathbf{I} \tag{41}$$

□

Finally, we derive the upper bound on the estimation error of the optimal estimator as follows.

Theorem A.3 (Frequency-Domain Error Bound of the Optimal Estimator)**.** *Under the assumptions in Sec. 4.2,, the optimal estimation errors in the low- and high-frequency components satisfy, respectively:*

$$e_L = \frac{1}{n}\mathbb{E}\|\mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} \tag{42}$$

$$e_H = \frac{1}{n}\mathbb{E}\|\mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \leq \frac{\eta}{n}\,\mathrm{Tr}(\mathbf{\Sigma}) \tag{43}$$

where n is the number of frames in the predicted trajectory, e_L and e_H denote the mean squared errors (MSEs) of the low- and high-frequency components, respectively. Consequently, the total error satisfies:

$$e = \frac{1}{n}\mathbb{E}\|x_0 - \mathbb{E}[x_0|x_t]\|^2 = \frac{1}{n}\mathbb{E}\|y_0 - \mathbb{E}[y_0|y_t]\|^2 = e_L + e_H \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} + \frac{\eta}{n}\,\mathrm{Tr}(\mathbf{\Sigma}) \tag{44}$$

Normalizing by the average per-frame signal energy, the relative error is bounded by:

$$\hat{e} = \frac{e}{\mathrm{Tr}(\mathbf{\Sigma})/n} \leq \frac{m}{\mathrm{Tr}(\mathbf{\Sigma})}\frac{\sigma_t^2}{\alpha_t^2} + \eta \tag{45}$$

Proof. We first consider the low-frequency error e_L:

$$e_L = \frac{1}{n}\mathbb{E}\|\mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \tag{46}$$

$$= \frac{1}{n}\mathbb{E}\left[\left(y_0 - \mathbb{E}[y_0|y_t]\right)^\top \mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\right] \tag{47}$$

$$= \frac{1}{n}\,\mathrm{Tr}\left(\mathbf{P}_L\,\mathrm{Cov}[y_0|y_t]\right) \tag{48}$$

$$\leq \frac{1}{n}\frac{\sigma_t^2}{\alpha_t^2}\,\mathrm{Tr}\left(\mathbf{P}_L\mathbf{I}\right) \tag{49}$$

$$= \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} \tag{50}$$

We then turn to the high-frequency error:

$$e_H = \frac{1}{n}\mathbb{E}\|\mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \tag{51}$$

$$= \frac{1}{n}\mathbb{E}\left[\left(y_0 - \mathbb{E}[y_0|y_t]\right)^\top \mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\right] \tag{52}$$

$$= \frac{1}{n}\operatorname{Tr}\left(\mathbf{P}_H \operatorname{Cov}[y_0|y_t]\right) \tag{53}$$

$$\leq \frac{1}{n}\operatorname{Tr}\left(\mathbf{P}_H\mathbf{\Sigma}\right) \tag{54}$$

$$\leq \frac{\eta}{n}\operatorname{Tr}\left(\mathbf{\Sigma}\right) \tag{55}$$

Furthermore, the total error satisfies:

$$e = \frac{1}{n}\mathbb{E}\|y_0 - \mathbb{E}[y_0|y_t]\|^2 \tag{56}$$

$$= \frac{1}{n}\mathbb{E}\left[\left(y_0 - \mathbb{E}[y_0|y_t]\right)^\top \mathbf{I}\left(y_0 - \mathbb{E}[y_0|y_t]\right)\right] \tag{57}$$

$$= \frac{1}{n}\mathbb{E}\left[\left(y_0 - \mathbb{E}[y_0|y_t]\right)^\top \mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\right] + \frac{1}{n}\mathbb{E}\left[\left(y_0 - \mathbb{E}[y_0|y_t]\right)^\top \mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\right] \tag{58}$$

$$= \frac{1}{n}\mathbb{E}\|\mathbf{P}_L\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 + \frac{1}{n}\mathbb{E}\|\mathbf{P}_H\left(y_0 - \mathbb{E}[y_0|y_t]\right)\|^2 \tag{59}$$

$$= e_L + e_H \tag{60}$$

$$\leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} + \frac{\eta}{n}\operatorname{Tr}\left(\mathbf{\Sigma}\right) \tag{61}$$

□

Remark A.4. The above discussion can be extended to the more general case where the data are concentrated in any frequency band of the DCT spectrum.

A.2 Extension to Non-Gaussian Conditional Distributions

Let x_0 have a finite second moment, and consider $x_t = \alpha_t x_0 + \sigma_t \epsilon$ with $\alpha_t > 0$. Assume that the noise ϵ is independent of x_0, has zero mean, and has identity covariance. Then, we have the following proposition holds.

Proposition A.5 (Frequency-Domain Error Bound)**.** *Let $\mathbf{P}_L$ be an orthogonal projector of rank m, let $\mathbf{P}_H = \mathbf{I} - \mathbf{P}_L$, and assume*

$$\operatorname{Tr}(\mathbf{P}_H\mathbf{\Sigma}) \leq \eta \operatorname{Tr}(\mathbf{\Sigma}). \tag{62}$$

Then the estimation errors satisfy

$$e_L = \frac{1}{n}\mathbb{E}[\|\mathbf{P}_L(y_0 - y_0^*)\|^2] \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2}, \tag{63}$$

$$e_H = \frac{1}{n}\mathbb{E}[\|\mathbf{P}_H(y_0 - y_0^*)\|^2] \leq \frac{\eta}{n}\operatorname{Tr}(\mathbf{\Sigma}). \tag{64}$$

For $\operatorname{Tr}(\mathbf{\Sigma}) > 0$, their sum and its normalized form satisfy

$$e \leq \frac{m}{n}\frac{\sigma_t^2}{\alpha_t^2} + \frac{\eta}{n}\operatorname{Tr}(\mathbf{\Sigma}), \tag{65}$$

$$\hat{e} = \frac{e}{\operatorname{Tr}(\mathbf{\Sigma})/n} \leq \frac{m}{\operatorname{Tr}(\mathbf{\Sigma})}\frac{\sigma_t^2}{\alpha_t^2} + \eta. \tag{66}$$

Proof. Under squared error, the conditional mean minimizes the expected error of every projected target. In the low-frequency subspace, compare it with the feasible estimator $\mathbf{P}_L y_t/\alpha_t$:

$$\mathbb{E}[\|\mathbf{P}_L(y_0 - y_0^*)\|^2] \leq \mathbb{E}[\|\mathbf{P}_L(y_0 - y_t/\alpha_t)\|^2] \tag{67}$$

$$= \frac{\sigma_t^2}{\alpha_t^2}\mathbb{E}[\|\mathbf{P}_L\xi\|^2] = m\frac{\sigma_t^2}{\alpha_t^2}. \tag{68}$$

In the high-frequency subspace, compare it with the zero estimator:

$$\mathbb{E}[\|\mathbf{P}_H(y_0 - y_0^*)\|^2] \le \mathbb{E}[\|\mathbf{P}_H y_0\|^2] = \mathrm{Tr}(\mathbf{P}_H \mathbf{\Sigma}). \tag{69}$$

Dividing by n and adding the two inequalities finally proves the result. □

B Details of the Synthetic Data Experiments

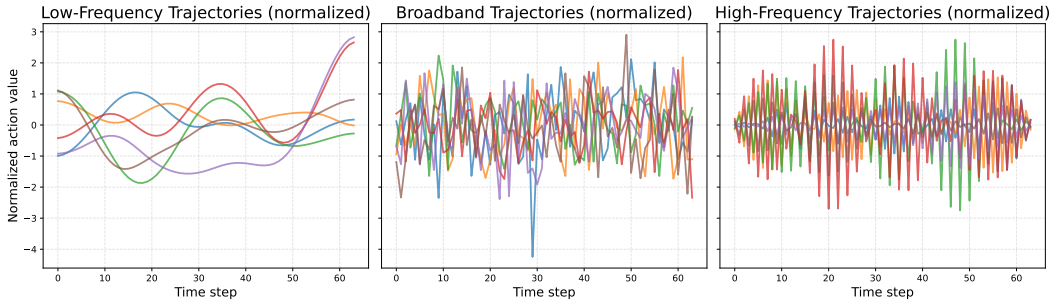


Figure 5: Examples of normalized synthetic trajectories from the low-frequency-dominant, broadband, and high-frequency-dominant data generators. For clarity, we plot the first action dimension only. Even in the standardized training space, `lowfreq` trajectories remain substantially smoother in the time domain, `broadband` trajectories exhibit richer multi-band variations, and `highfreq` trajectories oscillate rapidly across time.

B.1 Data Synthesis

We construct three synthetic trajectory datasets with controlled spectral concentration. For each sample, we first draw DCT-domain coefficients

$$z_{i,d,k} \sim \mathcal{N}(0, \lambda_k),$$

where i indexes the trajectory, d indexes the action dimension, and k indexes the DCT mode. We then transform the coefficients back to the time domain through the inverse DCT,

$$x_{i,d} = U^\top z_{i,d},$$

where U denotes the orthogonal DCT matrix. In this way, the training and sampling procedures are performed entirely in the time domain, while the DCT domain is used only to construct data with controlled spectra and to analyze the resulting errors.

We consider three data generators:

- `lowfreq`: the first six DCT modes are fixed as $(0.25, 1.00, 0.90, 0.65, 0.35, 0.20)$, while the remaining modes receive an exponentially decaying tail whose total energy is fixed to 1% of the full spectrum;
- `broadband`: $\lambda_k = 1$;
- `highfreq`: the last six DCT modes are fixed as $(0.25, 1.00, 0.90, 0.65, 0.35, 0.20)$, while the preceding modes receive an exponentially decaying prefix whose total energy is fixed to 1% of the full spectrum.

Under this construction, the first 6 modes account for 99% of the total energy in `lowfreq`, the last 6 modes account for 99% of the total energy in `highfreq`, and energy is uniformly distributed in `broadband`. For `lowfreq` and `broadband`, we define the 95% spectral cutoff by cumulative energy from low to high, which yields the first 6 modes for `lowfreq` and the first 61 modes for `broadband`. For `highfreq`, we instead define the dominant high-frequency band as the smallest suffix of DCT modes whose cumulative energy from high to low reaches 95%, which yields the last 6 modes. This construction provides a controlled contrast among strongly low-frequency-dominant, spectrally broad, and strongly high-frequency-dominant trajectories while keeping all other factors fixed. After synthesis, we normalize each dataset in the time domain on a per-action-dimension basis using the training-set mean and standard deviation computed over all samples and timesteps. All model training, sampling, and quantitative errors are computed in this standardized space. Fig. 5 presents examples of the synthetic data and illustrates that the `lowfreq` trajectories are significantly smoother, whereas the `highfreq` trajectories exhibit rapid oscillations.

B.2 Training Details

Each dataset contains 10,000 training trajectories and 256 test trajectories. All trajectories have length $n = 64$ and action dimension $d = 4$. We train a diffusion model directly in the time domain with the standard DDPM[10] ϵ-prediction objective, using the mean squared error loss.

The denoiser is a small 1D convolutional residual network. It uses a 1×1 input projection, 8 residual blocks with dilations $1, 2, 4, 8$ repeated cyclically, 64 hidden channels, sinusoidal timestep embeddings, and a timestep embedding dimension of 128. The model is optimized with Adam using a learning rate of 10^{-3}, a batch size of 256, and 60 training epochs.

For the forward diffusion process, we use 100 diffusion timesteps with a linear variance schedule from $\beta_{\min} = 10^{-4}$ to $\beta_{\max} = 2 \times 10^{-2}$. At inference time, we use deterministic DDIM[29] sampling with $\eta = 0$. To isolate the effect of the number of reverse updates, all step ablations share the same initial terminal noise. Following our main text, we treat the 100-step DDIM sample as the pseudo-truth trajectory and compare shorter DDIM chains against this reference.

We evaluate reverse-step counts $K \in \{1, \ldots, 100\}$. We further decompose the action-space error into its low-band and high-band components according to the 95% spectral cutoff of each dataset, in order to quantify how the low- and high-frequency errors evolve as the number of decoding steps changes. For `lowfreq` and `broadband`, the high-band is defined by the complement of the standard low-to-high 95% cutoff. For `highfreq`, the high-band is defined directly as the smallest suffix of DCT modes whose cumulative energy from high to low reaches 95%. In addition, we use the built-in controller in RoboTwin2.0[3] to execute the decoded trajectories and record the resulting tracking MSE, in order to examine whether the low-level controller can further suppress the impact of decoding errors.

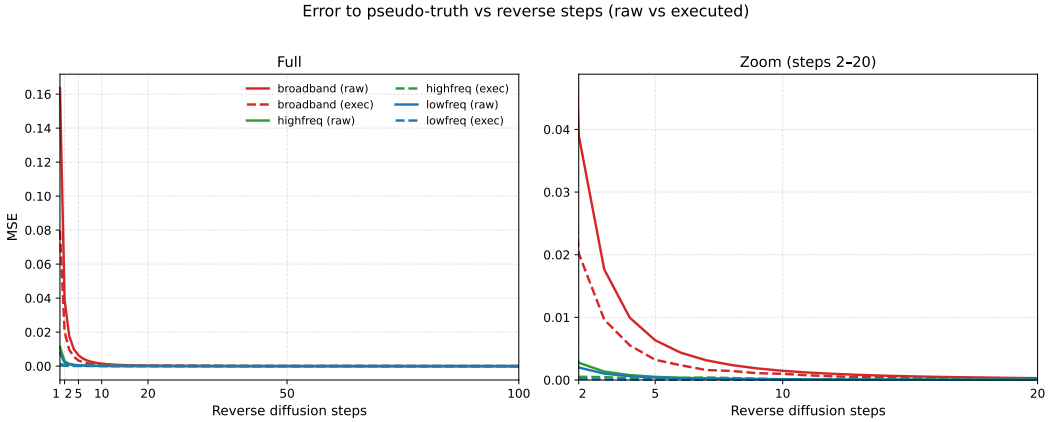


Figure 6: Decoding Error at Different Sampling Steps

B.3 Experimental Results

Table 8 and Figure 6 summarize the quantitative results for selected DDIM step counts. Both `lowfreq` and `highfreq` are spectrally concentrated in the DCT domain, and both therefore achieve substantially smaller reconstruction errors than `broadband`, whose energy is spread across all modes. At two reverse steps, the action-space MSE is 1.997×10^{-3} for `lowfreq` and 2.760×10^{-3} for `highfreq`, both far below the `broadband` value of 3.905×10^{-2}. At the same time, `lowfreq` remains consistently easier than `highfreq`. This is consistent with the spectral bias of neural networks[20], namely that low-frequency components are fitted more easily than high-frequency ones. As a result, although both datasets are spectrally concentrated, concentration in the low-frequency subspace is more favorable for few-step denoising than concentration in rapidly oscillating modes.

A further difference appears in the structure of the residual error. On `lowfreq`, the error drops sharply within the first few reverse updates: the action-space MSE decreases from 7.805×10^{-3} at one step to 1.997×10^{-3} at two steps, and further to 4.008×10^{-4} at five steps. Moreover, the residual few-step error is already concentrated in the high-frequency band: at two steps, the low-band MSE is only 3.598×10^{-4}, whereas the high-band MSE remains 1.637×10^{-3}. This gap becomes

Dataset	Steps	Action MSE	Low-band MSE	High-band MSE	Exec MSE	Exec Low-band MSE	Exec High-band MSE
lowfreq	1	0.007805	1.333×10^{-3}	0.006472	4.860×10^{-4}	4.141×10^{-4}	7.188×10^{-5}
lowfreq	2	0.001997	3.598×10^{-4}	0.001637	1.475×10^{-4}	1.203×10^{-4}	2.714×10^{-5}
lowfreq	5	4.008×10^{-4}	6.205×10^{-5}	3.388×10^{-4}	4.208×10^{-5}	3.557×10^{-5}	6.512×10^{-6}
lowfreq	10	9.587×10^{-5}	1.445×10^{-5}	8.142×10^{-5}	1.498×10^{-5}	1.302×10^{-5}	1.957×10^{-6}
lowfreq	50	1.175×10^{-6}	1.866×10^{-7}	9.880×10^{-7}	2.161×10^{-7}	1.848×10^{-7}	3.124×10^{-8}
lowfreq	100	0	0	0	0	0	0
broadband	1	0.1635	0.1562	7.382×10^{-3}	0.07955	0.07955	6.317×10^{-8}
broadband	2	0.03905	0.03729	1.760×10^{-3}	0.02022	0.02022	7.184×10^{-8}
broadband	5	6.339×10^{-3}	6.054×10^{-3}	2.852×10^{-4}	3.228×10^{-3}	3.228×10^{-3}	5.083×10^{-8}
broadband	10	1.467×10^{-3}	1.401×10^{-3}	6.592×10^{-5}	9.902×10^{-4}	9.901×10^{-4}	3.150×10^{-8}
broadband	50	1.855×10^{-5}	1.771×10^{-5}	8.330×10^{-7}	9.496×10^{-5}	9.495×10^{-5}	1.308×10^{-8}
broadband	100	0	0	0	0	0	0
highfreq	1	0.011386	0.010280	1.106×10^{-3}	0.001351	0.001351	1.676×10^{-8}
highfreq	2	0.002760	0.002461	2.987×10^{-4}	5.171×10^{-4}	5.170×10^{-4}	1.811×10^{-8}
highfreq	5	5.073×10^{-4}	4.527×10^{-4}	5.460×10^{-5}	2.490×10^{-4}	2.490×10^{-4}	1.958×10^{-8}
highfreq	10	1.152×10^{-4}	1.022×10^{-4}	1.294×10^{-5}	1.253×10^{-4}	1.253×10^{-4}	1.627×10^{-8}
highfreq	50	1.349×10^{-6}	1.181×10^{-6}	1.682×10^{-7}	7.722×10^{-6}	7.718×10^{-6}	3.717×10^{-9}
highfreq	100	0	0	0	0	0	0

Table 8: Few-step DDIM decoding errors measured against the 100-step pseudo-truth trajectory. We report both raw action-space errors and errors after RoboTwin's strict TOPP-based execution map. For highfreq, the high-band is defined as the smallest suffix of DCT modes whose cumulative energy from high to low reaches 95%.

even more pronounced after applying RoboTwin's TOPP-based execution map. For lowfreq, the executed-space MSE at two steps is reduced from 1.997×10^{-3} in action space to 1.475×10^{-4}, and the executed high-band MSE drops from 1.637×10^{-3} to only 2.714×10^{-5}.

By contrast, broadband still requires substantially more reverse updates: even after the same execution map is applied, its two-step executed MSE remains 2.022×10^{-2}. The highfreq control lies between the two extremes. Under the high-frequency-end 95% cutoff, its two-step error decomposes into a low-band MSE of 2.461×10^{-3} and a high-band MSE of only 2.987×10^{-4}, while the executed low-band and high-band MSEs are 5.170×10^{-4} and 1.811×10^{-8}, respectively. Thus, even after isolating the dominant high-frequency suffix, the residual few-step error in highfreq is not concentrated in that band as strongly as the lowfreq residual is concentrated in its high-frequency tail. Overall, these observations show that spectral concentration alone is not sufficient: few-step saturation depends specifically on concentration in low-frequency, task-relevant modes, while residual high-frequency errors are especially easy to suppress by the downstream low-pass execution pipeline.

C Implementation Details

C.1 Details of Simulation Experiment Settings

In this section, we provide the essential hyperparameters required to reproduce our results. Detailed hyperparameter choices are listed in Tab. 9.

Table 9: Hyperparameter Settings

Category	Hyperparameter	Value
Training		
	Batch size (Robotwin2.0)	256
	Batch size (Adroit & MetaWorld)	128
	Num. epochs	3000
	Optimizer	AdamW
	Weight decay(Robotwin2.0)	1×10^{-6}
	Weight decay(Adroit & MetaWorld)	1×10^{-8}
	LR scheduler	Cosine
	LR warmup steps	500
	Learning rate	1×10^{-4}
	Horizon (Robotwin2.0)	8
	Horizon (Adroit & MetaWorld)	16
	Num. action steps (Robotwin2.0)	6
	Num. action steps (Adroit & MetaWorld)	8
	Observation steps (Robotwin2.0)	3
	Observation steps (Adroit & MetaWorld)	2
	Encoder output dim.	64
	Diffusion timestep dim.	64
	MLP expansion ratio.	4
Inference		
	Num. inference steps	2
	Num. train steps	100

C.2 Details of Real-World Experiment Settings

The real-world experimental setup is shown in Fig. 7, and the task workflow is illustrated in Fig. 8. We design three tasks: Place Object, Adjust Bottle, and Stack Blocks Two. Specifically, the three tasks proceed as follows:

- `Place Object`: The robot picks up the target object and places it onto the designated platform.
- `Adjust Bottle`: The robot grasps the fallen cup and then sets it upright.
- `Stack Blocks Two`: The robot grasps one block and stacks it on top of another block.

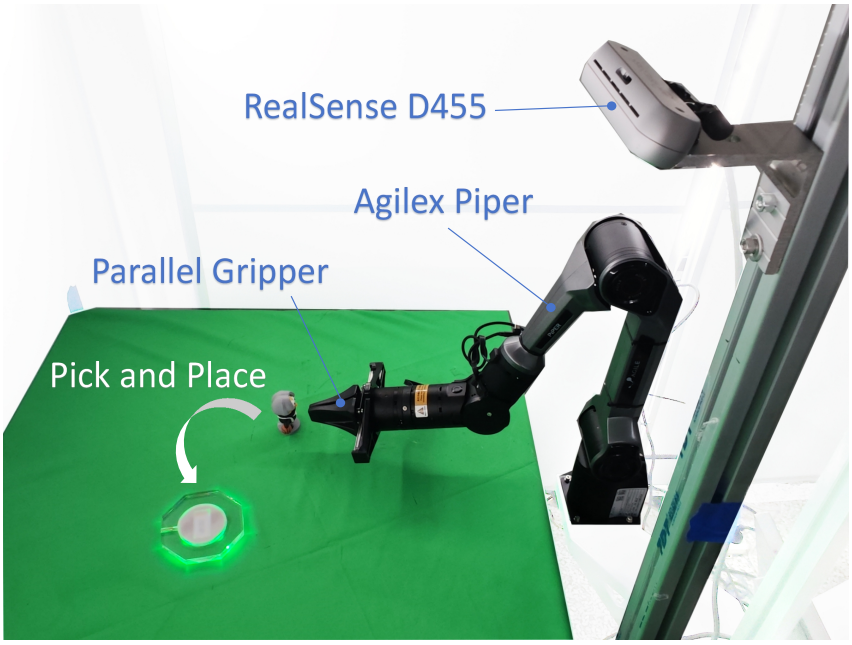


Figure 7: **Real-World Experiment Setup.**

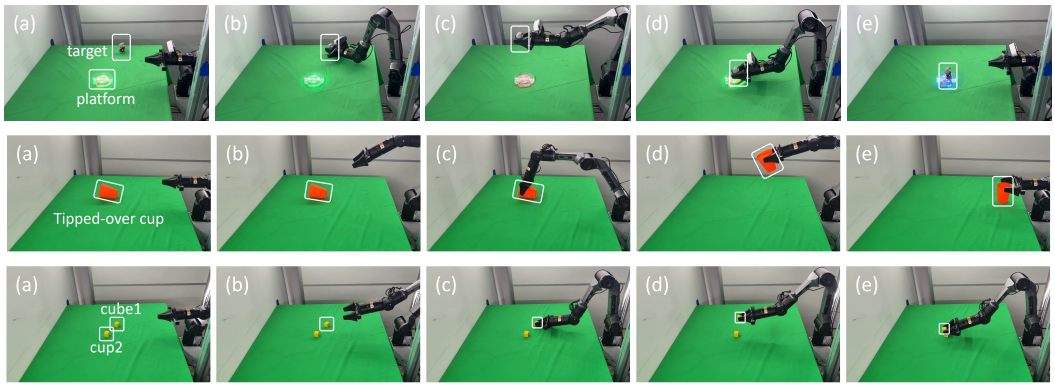


Figure 8: **Real-world Experiments.** The image sequence (top to bottom) illustrates the robot successfully performing three tasks: placing an object, uprighting a fallen cup, and stacking two blocks.

D Frequency-domain Decomposition Details

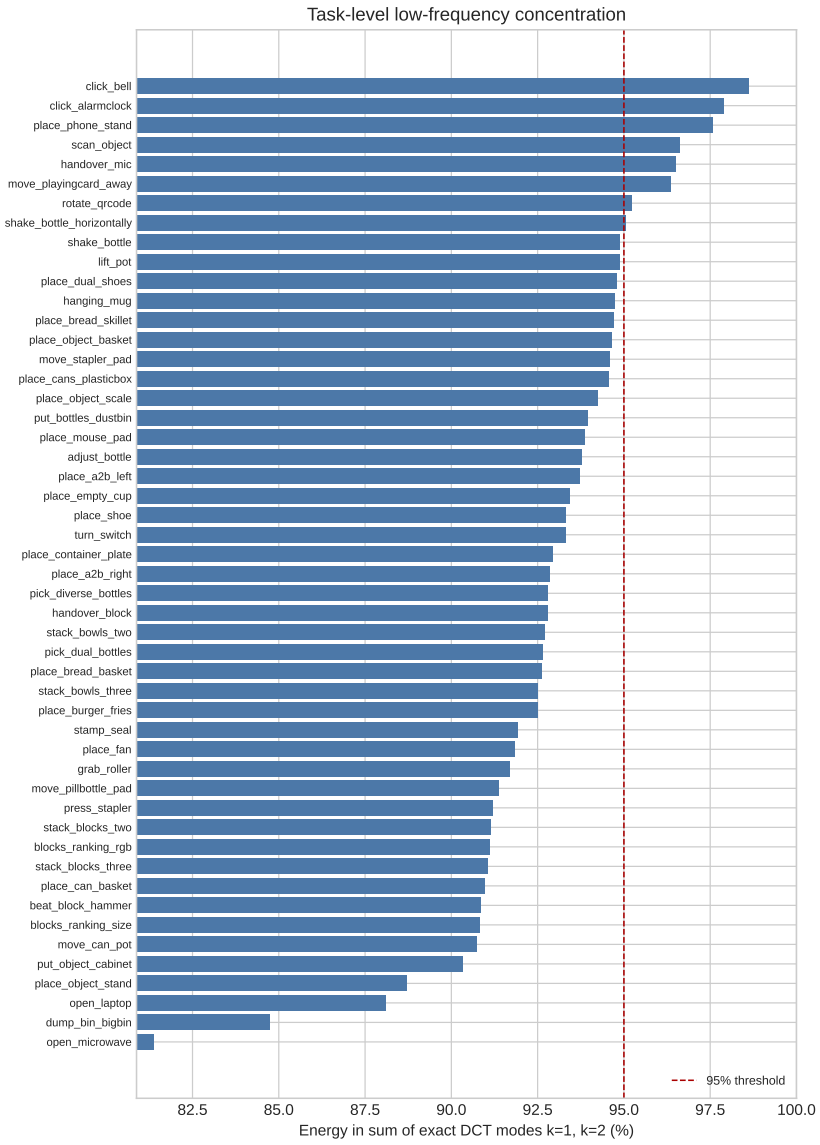


Figure 9: Fraction of Energy Contained in the First 5% of DCT Modes for Each Task

For each trajectory segment, we analyze the 14-dimensional action sequence $X \in \mathbb{R}^{T\times 14}$, where $X_{t,d}$ denotes the d-th action component at time step t. Each episode is partitioned into non-overlapping windows of length T (e.g., $T = 32$ or $T = 8$), and any remainder shorter than T is discarded. For each segment, we first remove the per-dimension temporal mean,

$$\tilde{X}_{t,d} = X_{t,d} - \frac{1}{T}\sum_{\tau=0}^{T-1} X_{\tau,d},$$

We then apply a one-dimensional DCT-II independently along the time axis of each action dimension, using orthonormal normalization:

$$C_{k,d} = \alpha_k \sum_{t=0}^{T-1} \tilde{X}_{t,d} \cos\left[\frac{\pi}{T}\left(t+\frac{1}{2}\right)k\right], \qquad k = 0,\ldots,T-1,$$

with $\alpha_0 = T^{-1/2}$ and $\alpha_k = (2/T)^{1/2}$ for $k \geq 1$. Because this transform is orthonormal, Parseval's identity holds, and the squared coefficients can be interpreted directly as energy. We therefore define the energy of mode k for a segment as

$$E_k = \sum_{d=1}^{14} C_{k,d}^2,$$

and its normalized energy share as

$$p_k = \frac{E_k}{\sum_{j=0}^{T-1} E_j}.$$

Dataset-level spectra are obtained by summing E_k over all segments and then normalizing by the total summed energy. We report both exact mode-wise energy shares $\{p_k\}_{k=0}^{T-1}$ in Sec. 4.1 and fraction of energy contained in the first 5%($k = 1, 2$) of DCT modes for each task in Fig. 9. As can be seen, the first two DCT modes account for more than 80% of the total energy for every task. Among the 50 tasks, only 4 have a cumulative energy ratio below 90% for these two modes. On average, the (k=1) mode alone accounts for 93.2% of the total energy, indicating that robot trajectories are highly concentrated in low-frequency components.

E Detailed Experimental Results on RoboTwin2.0

As shown in Table 10, we report the comparison of model success rates across 50 tasks on RoboTwin2.0.

	Adjust Bottle	Beat Block Hammer	Blocks Ranking RGB	Blocks Ranking Size	Click Alarmclock	Click Bell	Dump Bin Bigbin	Grab Roller	Handover Block	Handover Mic
DP	97.0	42.0	0.0	1.0	61.0	54.0	49.0	98.0	10.0	53.0
DP3	99.0	72.0	**3.0**	2.0	77.0	90.0	**85.0**	98.0	70.0	**100.0**
Flow Policy	**100.0**	75.0	0.0	0.0	**96.0**	**100.0**	80.0	98.0	16.0	67.0
MP1	95.0	68.0	2.0	3.0	91.0	98.0	83.0	**100.0**	**77.0**	93.0
HDP3 **(Ours)**	92.0	**88.0**	**3.0**	**8.0**	**96.0**	**100.0**	84.0	**100.0**	70.0	**100.0**

	Hanging Mug	Lift Pot	Move Can Pot	Move Pillbottle Pad	Move Playingcard Away	Move Stapler Pad	Open Laptop	Open Microwave	Pick Diverse Bottles	Pick Dual Bottles
DP	8.0	39.0	39.0	1.0	47.0	1.0	49.0	5.0	6.0	24.0
DP3	17.0	**97.0**	70.0	41.0	68.0	**12.0**	82.0	61.0	52.0	60.0
Flow Policy	6.0	48.0	40.0	14.0	64.0	1.0	76.0	17.0	64.0	83.0
MP1	11.0	81.0	87.0	35.0	**76.0**	5.0	70.0	35.0	**73.0**	**93.0**
HDP3 **(Ours)**	**35.0**	95.0	**95.0**	**56.0**	73.0	10.0	**85.0**	**92.0**	71.0	90.0

	Place A2B Left	Place A2B Right	Place Bread Basket	Place Bread Skillet	Place Burger Fries	Place Can Basket	Place Cans Plasticbox	Place Container Plate	Place Dual Shoes	Place Empty Cup
DP	2.0	13.0	14.0	11.0	72.0	18.0	40.0	41.0	8.0	37.0
DP3	46.0	**49.0**	26.0	19.0	72.0	67.0	48.0	86.0	13.0	65.0
Flow Policy	37.0	22.0	18.0	30.0	55.0	17.0	18.0	80.0	7.0	58.0
MP1	48.0	29.0	**40.0**	39.0	84.0	58.0	86.0	**87.0**	**14.0**	83.0
HDP3 **(Ours)**	**50.0**	27.0	39.0	**40.0**	**90.0**	**78.0**	**96.0**	**87.0**	8.0	**89.0**

	Place Fan	Place Mouse Pad	Place Object Basket	Place Object Scale	Place Object Stand	Place Phone Stand	Place Shoe	Press Stapler	Put Bottles Dustbin	Put Object Cabinet
DP	3.0	0.0	15.0	1.0	22.0	13.0	23.0	6.0	22.0	42.0
DP3	36.0	4.0	**65.0**	**15.0**	60.0	44.0	**58.0**	69.0	60.0	**72.0**
Flow Policy	9.0	2.0	21.0	6.0	51.0	44.0	39.0	**92.0**	2.0	7.0
MP1	27.0	2.0	53.0	14.0	61.0	52.0	40.0	64.0	25.0	48.0
HDP3 **(Ours)**	**39.0**	**7.0**	**65.0**	9.0	**65.0**	**65.0**	57.0	73.0	**64.0**	63.0

	Rotate QRcode	Scan Object	Shake Bottle	Shake Bottle Horizontally	Stack Blocks Three	Stack Blocks Two	Stack Bowls Three	Stack Bowls Two	Stamp Seal	Turn Switch
DP	13.0	9.0	65.0	59.0	0.0	7.0	63.0	61.0	2.0	36.0
DP3	**74.0**	31.0	98.0	**100.0**	1.0	24.0	57.0	83.0	18.0	46.0
Flow Policy	14.0	35.0	98.0	93.0	0.0	13.0	0.0	73.0	20.0	50.0
MP1	44.0	41.0	97.0	98.0	1.0	34.0	50.0	77.0	28.0	**62.0**
HDP3 **(Ours)**	55.0	**45.0**	**100.0**	**100.0**	**2.0**	**41.0**	**74.0**	**89.0**	**43.0**	55.0

Table 10: Evaluation on the Robotwin2.0 benchmark. Each task is tested across 100 randomly generated scenes.

F Additional Comparisons

F.1 Comparison with Frequency-Aware Policies

We further compare HDP3 with FreqPolicy-AR[41] and FreqPolicy-Flow[31] on four tasks from Adroit and MetaWorld. The results are shown in Tab. 11. HDP3 achieves the highest average success rate while using fewer parameters and lower inference latency.

Table 11: **Comparison with frequency-aware policies.** Success rates are reported as SR_5 (%). All latencies are measured on the same RTX 5880 Ada GPU.

Method	Params	Latency	Door	Pen	Disassemble	Hand-Insert	Avg.
FreqPolicy-AR	63.40	68.68	**62.3**	47.3	80.7	30.3	55.2
FreqPolicy-Flow	255.80	7.06	58.5	44.0	78.2	17.7	49.6
HDP3 **(Ours)**	**2.52**	**4.50**	57.3	**48.7**	**87.7**	**33.3**	**56.8**

F.2 Comparison across 2D, 3D, and 4D Representations

To examine the effects of observation representation and decoder design, we compare DiM and U-Net decoders with RGB/2D, point-cloud/3D, and spatiotemporal/4D encoders. The 4D results use a simplified reference implementation based on DP4[12]. As shown in Tab. 12, the 3D and 4D representations achieve higher average success rates than the 2D representation. Replacing the U-Net decoder with DiM also improves the average performance within each representation, supporting the applicability of the DiM decoder across different observation representations.

Table 12: **Encoder–decoder comparisons on Adroit.**

Encoder	Decoder	Door	Hammer	Pen
RGB/2D	DP U-Net	44.7	91.3	10.0
RGB/2D	DiM	52.0	98.0	52.0
Point cloud/3D	DP3 U-Net	62.0	100.0	43.7
Point cloud/3D	DiM	57.3	100.0	48.7
Spatiotemporal/4D	U-Net	59.3	100.0	48.0
Spatiotemporal/4D	DiM	65.0	100.0	48.3

F.3 Small-Model Baselines and Training Objectives

We further compare HDP3 with small variants of DP3, Flow Policy, and MP1. We also train the HDP3 architecture with a flow-matching objective, denoted by HDP3-FM. The results are shown in Tab. 13. HDP3 outperforms the small-model baselines on all three tasks, while HDP3-FM achieves comparable performance to HDP3.

Table 13: **Small-model baselines and flow-matching variant.**

Method	Door	Hammer	Pen
HDP3	**57.3**	**100.0**	**48.7**
HDP3-FM	56.7	**100.0**	47.8
Small-DP3	28.5	91.3	42.8
Small-Flow Policy	33.7	16.3	31.7
Small-MP1	0.0	3.7	10.3

G Extension to RGB-Based Observations

We further apply the DiM decoder to an RGB-conditioned diffusion policy, denoted by DP-DiM. The results on Adroit and MetaWorld are shown in Tabs. 14 and 15, respectively. Replacing the original decoder with DiM improves the average success rate from 48.3% to 67.3% on Adroit and from 20.6% to 32.4% on the selected MetaWorld tasks. These results show that the DiM decoder is also effective for RGB-based visuomotor policies.

Table 14: **RGB-conditioned policies on Adroit.**

Method	Door	Hammer	Pen	Avg.
DP	44.0	91.0	10.0	48.3
DP-DiM	**52.0**	**98.0**	**52.0**	**67.3**

Table 15: **RGB-conditioned policies on MetaWorld.**

Task	DP	DP-DiM
Assembly	3.3	**62.0**
Disassemble	13.3	**30.7**
Hand-Insert	7.5	**9.0**
Pick-Place-Wall	6.7	**12.0**
Push	3.3	**6.3**
Reach-Wall	**48.3**	47.0
Stick-Push	**61.7**	59.7
Avg.	20.6	**32.4**

H Limitations

H.1 Spectral Differences across Embodiments

The frequency structure of action trajectories can vary across robot embodiments. We further analyze datasets spanning dexterous-hand, bimanual, tactile, mobile, and whole-body humanoid manipulation, as shown in Tab. 16. Standard bimanual and mobile-manipulation trajectories exhibit strong low-frequency concentration. The two contact-rich RoboTwin2.0 tasks also retain 96.87% of their energy in the first two DCT modes. In contrast, dexterous-hand actions have broader spectra: these modes account for only 54.74% of the energy on Adroit. Similarly, the arm actions in SIMPLE remain low-frequency dominated, while its hand actions exhibit a broader spectrum.

This difference is consistent with our empirical results. HDP3's advantage is substantially reduced on Adroit, where its performance is nearly on par with DP3, while the improvement remains pronounced on the two low-frequency-dominated RoboTwin2.0 tasks. Therefore, the low-frequency assumption and the resulting benefit of compact decoding can depend on the embodiment and its action spectrum.

Table 16: **Action spectra across embodiments.** Energy fractions and policy scores are in percent. "–" indicates that policy performance was not evaluated.

Embodiment	Benchmark	Energy in $k = 1, 2$	DP3	HDP3
Dexterous hand	Adroit	54.74	68.6	68.7
Bimanual contact-rich	RoboTwin2.0	96.87	71.5	88.5
Tactile manipulation	UniVTAC	96.50	–	–
Mobile bimanual	BiGym	93.26	–	–
Whole-body humanoid	SIMPLE (overall)	77.10	–	–
Dexterous hands	SIMPLE (hands)	76.28	–	–
Arm manipulation	SIMPLE (arms)	96.57	–	–

H.2 Fine-Grained and Directionally Controlled Manipulation

To further evaluate tasks involving abrupt contact transitions, high-speed motions, dexterous manipulation, or discontinuous action changes, we select 13 additional MetaWorld tasks. The results are shown in Tab. 17. Although HDP3 still outperforms DP3 on average, it exhibits noticeable performance drops on Dial Turn, Lever Pull, and Peg Insert Side, which require precise, directionally controlled manipulation. These results indicate that HDP3 may be less effective on tasks requiring fine-grained control and may fail to execute such precise manipulation reliably.

Table 17: **Additional MetaWorld tasks.** Scores are in percent. The average is the arithmetic mean of the 13 displayed task scores.

Task	DP3	HDP3
Basketball	**100.0**	**100.0**
Bin Picking	19.0	**57.0**
Push Wall	77.0	**86.3**
Dial Turn	**82.0**	69.5
Handle Pull	32.0	**43.0**
Lever Pull	**79.0**	62.0
Door Unlock	**100.0**	**100.0**
Peg Insert Side	**85.0**	63.3
Hand Insert	14.0	**37.0**
Pick Out of Hole	16.0	**30.0**
Hammer	**100.0**	**100.0**
Soccer	18.0	**23.0**
Sweep Into	15.0	**43.0**
Average	56.7	**62.6**

I Broader Impacts

This work studies how to make diffusion-based visuomotor policies smaller and faster while maintaining strong manipulation performance. A potential positive impact is that more efficient policies can lower the hardware and energy cost of robotics research and deployment, making strong visuomotor control methods more accessible to smaller labs and resource-constrained platforms. In addition, faster inference can improve the practicality of closed-loop robot control in real-world manipulation settings where latency matters.

At the same time, more efficient manipulation policies can also introduce risks if they are deployed without adequate safeguards. In particular, visuomotor policies may fail under distribution shift, unexpected contacts, sensor noise, or task conditions that differ from those seen in training. Such failures can lead to unsafe motions, unintended collisions, or damage to surrounding objects. More broadly, improved manipulation capability may accelerate automation in labor-sensitive settings or be adapted to applications that require stronger safety, reliability, or accountability guarantees than those studied here.

We therefore emphasize that our experiments are limited to benchmark environments and controlled laboratory settings, and we do not claim that the method is ready for safety-critical deployment. We encourage future users to combine policies of this kind with standard robotic safety measures, including simulation-based validation before deployment, conservative workspace and collision constraints, human oversight during testing, and hardware-level protections such as emergency stops and motion limits.

J External Assets, Licenses, and Planned Release

Our experiments rely on several existing benchmarks and open-source codebases. For RoboTwin2.0, we use the official RoboTwin repository and benchmark assets; the official repository is released under the MIT license. For MetaWorld, we use the official MetaWorld benchmark implementation, which is also released under the MIT license. For Adroit, we follow the standard D4RL/Adroit benchmark assets; the D4RL codebase is released under the Apache-2.0 license, and the benchmark maintainers state that, unless otherwise noted, the datasets are released under CC BY 4.0.

For baseline implementations, we use official open-source code when available and cite the corresponding original papers in the bibliography. In particular, the official public repositories for Diffusion Policy, 3D Diffusion Policy (DP3), and FlowPolicy are released under the MIT license. We follow the documented licenses and terms of use for these assets and will include the exact repository versions, package versions, or commit identifiers used in our experiments in the supplemental material.

To support reproducibility, we will provide anonymized code and documentation alongside the submission's supplemental material. The release package will include environment setup instructions, exact commands for training and evaluation, and a license file for the released code.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main claims: low-frequency structure in robot trajectories, a theory motivating few-step denoising, a lightweight DiM decoder, and strong empirical performance. These claims are supported by the theoretical analysis in Sec. 4.2, the synthetic experiments in Sec. 5, and the benchmark and real-world results in Sec. 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly points to a limitations discussion in the conclusion, and the limitations are summarized in App. H. In particular, the paper notes that validation is currently focused on 3D-input visuomotor policies and that the theoretical analysis could be further strengthened.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions used for the frequency-domain analysis are stated in Sec. 4.2, including the DCT-domain formulation, Gaussian assumption, and low-/high-frequency decomposition. The main result is stated as Theorem 4.1, and detailed supporting lemmas and proofs are provided in App. A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the training and evaluation protocols, benchmark/task setup, number of demonstrations, seeds, optimization settings, DDIM schedule, ablations, and key implementation hyperparameters in Secs. 5 and 6 and App. B and App. C. These details are the ones that directly affect the main empirical claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide anonymized code and reproduction instructions in the supplemental material to support faithful reproduction of the main experimental results. App. J summarizes the planned release contents, including environment setup instructions, exact commands, and license information for the released code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 6 describes the benchmark protocols, data regimes, evaluation metrics, training schedules, optimizer, inference steps, and normalization strategy. App. C further lists the essential hyperparameters used in training and inference.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The paper reports averages over multiple seeds and evaluation episodes, but it does not include error bars, confidence intervals, or formal statistical significance tests for the reported results. The current tables therefore summarize central tendency only.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper reports the main hardware used, including an NVIDIA RTX 5880 Ada GPU for training/evaluation and an RTX 4060 GPU for real-world deployment, and it reports inference latency in Tab. 3. However, it does not provide full training-time, memory, or total-compute estimates for each experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the research conforms to the NeurIPS Code of Ethics. The work uses standard robotic benchmarks together with controlled real-world robot experiments, and the submission preserves anonymity.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: App. I discusses both positive impacts, such as improved efficiency and accessibility of robotic learning systems, and negative impacts, such as unsafe deployment, automation-related concerns, and misuse outside the tested regime. It also outlines mitigation measures including simulation-first validation, human oversight, and hardware safety constraints.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not describe the release of a high-risk foundation model, scraped internet-scale dataset, or similarly dual-use artifact that would require special access controls or misuse safeguards. The work centers on visuomotor control policies evaluated in standard robotics settings.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: App. J explicitly documents the external benchmarks and codebases used in the paper, including RoboTwin2.0, MetaWorld, D4RL/Adroit, Diffusion Policy, DP3, and FlowPolicy, together with their licenses or terms of use. The manuscript also cites the original papers for these assets in the bibliography.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release anonymized code for the proposed method and provide documentation and reproduction instructions alongside the released assets in the supplemental material. App. J summarizes the planned release contents, including setup instructions, exact commands, and license information for the released code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing studies or experiments on human participants. The teleoperated demonstrations described for the real-world setup are used as expert control data rather than as a human-subject study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The manuscript does not report crowdsourcing or human-subject experiments that would require participant-risk disclosure or IRB-style review. Accordingly, this item is not applicable to the present submission.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Any LLM usage was limited to writing, language polishing, or formatting assistance and did not affect the core methodology, scientific rigor, or originality of the research. Therefore, under the NeurIPS checklist instructions, declaration in this item is not required.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.