1STEP-WAM: YOUR WORLD ACTION MODEL BUT FASTER USING 1-STEP ACTION GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) unify visual world modeling and action generation, yet their real-world deployment is heavily bottlenecked by iterative action denoising and the sequential execution of visual prefilling and action prediction. We demonstrate that the action-denoising trajectories in WAMs are approximately straight, rendering the first-step velocity a strong estimate of the final action direction. Building on this observation, we introduce 1Step-WAM, a framework that transforms multi-step action denoising into one single forward. Specifically, we employ a lightweight calibrator to directly correct the first step velocity, and use bandwidth-constrained action decoding to further reduce errors in the calibrated one-step actions. Compressing action generation into a single step enables it to execute in parallel with visual prefilling, completely breaking the sequential bottleneck of the traditional imagination-then-action paradigm. Evaluations on RoboTwin and LIBERO show that 1Step-WAM largely preserves multi-step task success with only about 13.4M trainable parameters, while achieving inference speedups of $7.44\times$, $5.98\times$, and $24.84\times$ on RoboTwin for FastWAM, ImageWAM and Kairos, respectively. Furthermore, real-world robotic deployments using the FastWAM backbone confirm a consistent $7.44\times$ speedup while maintaining close success rate to the multi-step baseline.

1 INTRODUCTION

In the field of Embodied AI, making robot policy capable of generating right actions across diverse environments and scenarios remains a long-standing core objective. In the past two years, Vision-Language-Action (VLA) models (Kim et al., 2024; Brohan et al., 2023; Black et al., 2024; Physical Intelligence et al., 2025; Bjorck et al., 2025; Shukor et al., 2025; Team et al., 2025; Team, 2025)have become a widely used model architecture in Embodied AI, it utilizes a pretrained vision-language-model to generate the actions from current visual observations and language instructions. However, VLA models adopt the static VLM visual understanding ability, it lacks the ability of understand how the physical world will change under interaction, limiting their generalization ability. This has motivated the emergence of WAMs (Yuan et al., 2026; Zhang et al., 2026; Li et al., 2026a), which unify future visual world prediction and action generation in a single model, exhibiting better generalization ability than VLAs.

Despite WAMs' superior generalization, the real-world deployment of WAMs is significantly bottlenecked by inference latency. The standard WAM pipeline relies on a sequential imagination-then-action process, they first imagine a future scence using a visual diffusion models (Wan et al., 2025; Hong et al., 2022; Labs, 2025; Ali et al., 2025) through multi step denoise, then denoise the actions based on the imagined future. While recent optimization efforts have successfully accelerated the imagination phase by reducing the video denoising process to a single-step visual prefill, they largely overlook the latency bottleneck within the action generation phase itself. In these existing frameworks, a visual Diffusion Transformer (DiT) (Peebles & Xie, 2023) first processes observations and instructions to extract visual KV caches, after which an action DiT must still undergo a time-consuming, multi-step iterative denoising process to produce control signals. This leaves two critical efficiency issues unresolved: the inherent latency of iterative action denoising, and the serial dependency that forces action generation to stall until the visual prefill is fully completed. This sequential mechanism fundamentally blocks the potential for parallel execution, thereby restricting the inference speed necessary for high-frequency robotic control (Black et al., 2026).

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

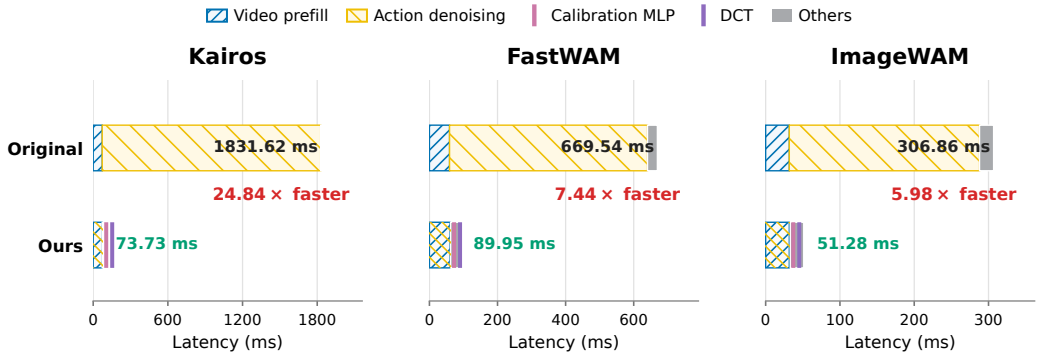


Figure 1: Speedups of 1Step-WAM over the original multi-step FastWAM, ImageWAM, and Kairos.

To address these challenges, we propose 1Step-WAM, a novel framework that realizes one-step, parallel action generation for WAMs. We find that when the visual observation, language instruction, and robot state are fixed, the target action is nearly invariant to the initial noise used for generation. Building on this observation, we theoretically characterize the denoising dynamics and show that the action trajectory is approximately straight: the first-step velocity provides a strong estimate of the final action direction, Exploiting this property, we design a lightweight calibrator attached to the action DiT. This calibrator learns to directly correct the initial predicted velocity toward the multi-step teacher trajectory, effectively transforming the computationally expensive iterative solver into a direct noise-to-action one-step policy. Furthermore, to mitigate the remaining action deviation between our one-step policy and the multi-step teacher, we analyze the generated actions in the frequency domain. Per-joint DCT visualizations in Appendix 7.2 show that teacher action concentrates most of their spectral energy in low-frequency modes. This observation motivates bandwidth-constrained action decoding: we apply a temporal DCT along the action horizon and retain only the first K modes. The projection trades the loss of high-frequency teacher components against the removal of prediction error in the discarded frequencies. Our analysis shows that this projection reduces the expected squared error relative to the teacher when the discarded prediction-error energy exceeds the discarded teacher-action energy. Most importantly, compressing the action generation into a single step unlocks a critical system-level breakthrough. Instead of waiting for the visual prefill to finish, 1Step-WAM integrates action generation directly into the visual prefilling computation, eliminating the sequential waiting time of the traditional imagination-then-action paradigm.

We comprehensively evaluate our proposed OneStep-WAM in both simulated and real-world tasks. To demonstrate its broad applicability, we apply our method to three WAM models: FastWAM, ImageWAM, and Kairos. On the RoboTwin simulation benchmark, our approach achieves remarkable speedups of $7.44\times$ for FastWAM, $5.98\times$ for ImageWAM, and $24.84\times$ for Kairos, as shown in Fig. 1, all while suffering a negligible drop in task success rate. Furthermore, we validate 1Step-WAM on real robot tasks using the FastWAM backbone. In these real-world deployments, our method consistently delivers a $7.44\times$ acceleration and maintains an average task success rate 75%, compared to the 80% of the original multi-step denoise baseline.

Our main contributions can be summarized as follows:

- We demonstrate that WAM action trajectories are approximately straight and highly invariant to initial noise. This establishes that the first-step predicted velocity inherently includes the primary direction of the final target action.
- We propose 1Step-WAM, a framework that combines lightweight velocity calibration, bandwidth-constrained action decoding, and joint video-action execution to accelerate WAM action generation.
- We evaluate 1Step-WAM across FastWAM, ImageWAM, and Kairos on RoboTwin and LIBERO, and on real-robot tasks with FastWAM, 1Step-WAM achieves end-to-end inference speedups of $7.44\times$, $5.98\times$, and $24.84\times$ on the three backbones respectively, while largely preserving task success.

2 Related Work

2.1 World Action Models

World Action Models (WAMs) use visual world modeling to support action generation. A central design choice is how predictive visual representations are used during inference. Recent approaches reduce the dependence on explicit future generation at deployment. Fast-WAM (Yuan et al., 2026) and GigaWorld-Policy retain visual-dynamics supervision during training while enabling inference without future-video synthesis (Ye et al., 2026). ImageWAM instead conditions action generation on features from an image-editing model (Zhang et al., 2026), while Kairos (Team et al., 2026) integrates understanding, generation, and prediction within a unified framework.

These approaches reduce the cost of visual world modeling, but action generation still require iterative denoising. We address this bottleneck by converting pretrained multi-step action generators into one-step policies. This approach complements efforts to reduce the inference cost of visual world modeling.

2.2 Diffusion Distillation

Diffusion distillation reduces the sequential network evaluations required for sampling. Representative approaches include progressive step reduction (Salimans & Ho, 2022), trajectory consistency (Song et al., 2023), distribution matching (Yin et al., 2024b;a), and step-size-conditioned shortcuts (Frans et al., 2025). These ideas also enable faster robot control, in robotics, Consistency Policy (Prasad et al., 2024) and OneDP (Wang et al., 2024) apply consistency and distribution matching to diffusion policies. One-Step Flow Policy uses self-distillation for flow-based control (Li et al., 2026b), while SnapFlow (Luan et al., 2026) progressively distills flow-matching VLAs into one-step action generators.

However, all these methods need to train the whole dit params which is expensive for training. Our method focuses on action generation in pretrained WAMs: we freeze the backbone and train only a lightweight head to calibrate the initial velocity toward the multi-step teacher endpoint.

3 Rethinking: Can a WAM Generate Actions in a Single Step?

In continuous action generation for WAMs, action dit typically require multi-step iterative denoising to generate an action chunk. However, we observe a critical phenomenon: when visual observations, language instructions and robot state are fixed, the final action distribution decoded through multiple steps exhibits extremely small variance, regardless of the initial noise at the starting point of denoising. we run FastWAM's multi-step denoiser with 10,000 independently sampled initial noises. The resulting actions have an empirical variance of 1.182×10^{-5}. This phenomenon inspires us to rethink: *Can a WAM generate usable actions in a single forward pass?*

To mathematically answer this question, we conduct a theoretical derivation from the trajectory geometry of Flow Matching (Lipman et al., 2022). For fixed conditioning inputs, we assume the clean target action $x_0 \sim \mathcal{N}(\mu, \sigma^2 I_d)$, with $\sigma > 0$, and the pure noise $x_1 \sim \mathcal{N}(0, I_d)$. The forward noising path is defined as $x_t = (1-t)x_0 + tx_1$, where $t \in [0, 1]$; denoising proceeds in the reverse direction, from $t = 1$ to $t = 0$. Since the clean target x_0 and the noise x_1 are sampled independently for this interpolation, the marginal variance of the intermediate state x_t is:

$$\text{Var}(x_t) = (1-t)^2\text{Var}(x_0) + t^2\text{Var}(x_1) = \left[(1-t)^2\sigma^2 + t^2\right] I_d \tag{1}$$

For clarity, we denote this time-dependent variance scalar as $D(t) = (1-t)^2\sigma^2 + t^2$. In the flow matching framework, the optimal vector field is defined with respect to increasing t as the conditional expectation of the interpolation velocity given the current state:

$$v^*(x_t, t) = \mathbb{E}[x_1 - x_0 \mid x_t] \tag{2}$$

Since the independently sampled endpoints are jointly Gaussian, this conditional expectation can be evaluated analytically. Using $\mathbb{E}[x_1 - x_0] = -\mu$, the marginal mean $\mathbb{E}[x_t] = (1-t)\mu$, the variance

$D(t)I_d$, and the cross-covariance $\mathrm{Cov}(x_1 - x_0, x_t) = [t - (1-t)\sigma^2]I_d$, the Gaussian conditional expectation identity gives:

$$\begin{aligned} v^*(x_t, t) &= -\mu + \frac{t - (1-t)\sigma^2}{D(t)}\big[x_t - (1-t)\mu\big] \\ &= -\frac{t}{D(t)}\mu + \frac{t - (1-t)\sigma^2}{D(t)}x_t \end{aligned} \tag{3}$$

During inference, the system integrates the ordinary differential equation (ODE) $\dot{x}_t = v^*(x_t, t)$ backward from $t = 1$ to $t = 0$, with initial noise $x(1) = x_1$. Here, $v^*(x, t)$ specifies the velocity at any state x, whereas $\dot{x}_t$ is the derivative of the particular trajectory with respect to increasing t; a denoising step uses $\Delta t < 0$. Below, x_t denotes this inference trajectory, which shares the marginals of the training interpolation but couples its endpoints through the ODE. By analytically solving this first-order linear ODE, we obtain:

$$x_t = \sqrt{D(t)}x_1 + (1-t)\mu = \sqrt{(1-t)^2\sigma^2 + t^2}\, x_1 + (1-t)\mu \tag{4}$$

At the clean endpoint, this gives $x_{t=0} = \mu + \sigma x_1$, which has the assumed target distribution $\mathcal{N}(\mu, \sigma^2 I_d)$.

Taking the derivative of this analytical trajectory with respect to t gives:

$$\dot{x}_t = -\mu + C(t)x_1, \quad \text{where } C(t) = \frac{t - (1-t)\sigma^2}{\sqrt{(1-t)^2\sigma^2 + t^2}} \tag{5}$$

The trajectory velocity exhibits two regimes as the conditional output variance approaches zero ($\sigma \to 0^+$):

- **Collinear Main Interval:** For any fixed $\epsilon \in (0, 1)$, on the time interval $t \in [\epsilon, 1]$, the time-dependent noise coefficient $C(t)$ uniformly converges to 1. Thus, $\dot{x}_t$ approaches the constant vector $x_1 - \mu$. Since denoising proceeds with decreasing t, its direction approaches $\mu - x_1$, and the trajectory approaches a straight line over this interval.
- **Terminal transition:** Within the narrow terminal interval $t = \mathcal{O}(\sigma)$ near the clean endpoint, the coefficient $C(t)$ departs from 1, reaching $C(0) = -\sigma$. Thus, $\dot{x}_{t=0} = -\mu - \sigma x_1 \to -\mu$ as $\sigma \to 0^+$; the reverse-time denoising velocity $-\dot{x}_t$ approaches μ at this endpoint.

The theoretical derivations above show that, under the small-variance Gaussian model, a velocity prediction at the noise endpoint $t = 1$ identifies the primary denoising direction $\mu - x_1$. With this time convention, a one-step Euler estimate is $\widehat{x}_0 = x_1 - v^*(x_1, 1)$, where the minus sign follows from the time step $\Delta t = -1$.

4 METHODS

4.1 LIGHTWEIGHT VELOCITY CALIBRATION

Motivated by the nearly linear denoising trajectory, we introduce a lightweight velocity calibration head to calibrate the initial velocity for one-step action prediction.

Calibration Inputs and Formulation. Let $x_1 \sim \mathcal{N}(0, I_d)$ denote the initial noise. In our one-step framework, we execute the frozen ActionDiT backbone only once at $t = 1$. This single forward pass yields the raw velocity prediction v_1, alongside the intermediate hidden features h_1 last action dit block output.

To compensate for the initial regression error inherent in predicting from pure noise, we introduce a MLP-based calibration head, denoted as f_ϕ. This head utilizes the priors informations to predict a velocity residual Δv:

$$\Delta v = f_\phi(h_1, x_1, v_1) \tag{6}$$

The calibrated effective velocity can be formulated as $v_{cal} = v_1 + \Delta v$. Consequently, the one-step action can be expressed as:

$$\hat{a}_0 = x_1 - (v_1 + f_\phi(h_1, x_1, v_1)). \tag{7}$$

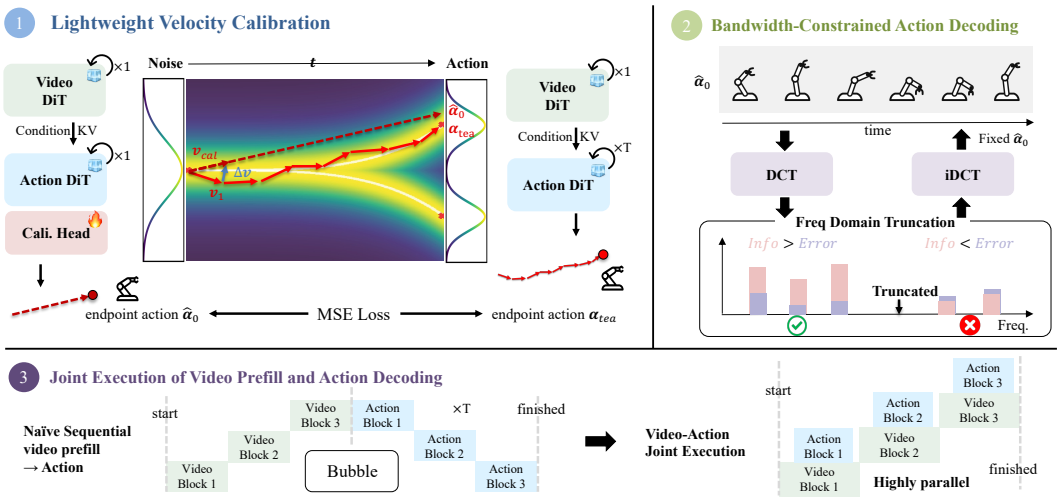


Figure 2: Overview of 1Step-WAM. The method combines lightweight velocity calibration, bandwidth-constrained action decoding, and joint execution of video prefill with one-step action generation.

where $\hat{a}_0$ represents the final generated action sequence.

Calibration Objective. The training objective for the calibration head is to minimize the discrepancy between the one-step generated action $\hat{a}_0$ and the multi-step teacher action a_{tea}. During training, we freeze the WAM backbone, and only update the calibration head parameters ϕ. The training objective is formulated as a Mean Squared Error (MSE) loss directly in the action space:

$$\mathcal{L} = \mathbb{E}\left[\|\hat{a}_0 - a_{tea}\|_2^2\right] \tag{8}$$

By minimizing $\mathcal{L}$, the calibration head implicitly learns the non-linear transformation required to map the raw initial velocity to an effective velocity that better matches the teacher endpoint.

4.2 Bandwidth-Constrained Action Decoding

Velocity calibration learns to approximate the multi-step teacher endpoint with a single ActionDiT forward. The endpoint-matching objective, however, does not explicitly constrain the temporal bandwidth of the predicted action chunk. We complement calibration with a fixed projection onto a low-frequency temporal subspace. This projection trades the loss of teacher motion in discarded frequencies against the removal of prediction error in those frequencies.

Temporal projection. Let $\widehat{A}_c \in \mathbb{R}^{T\times d_c}$ denote the calibrated one step action, and let $C \in \mathbb{R}^{T\times T}$ be the orthonormal DCT-II matrix. We define M_K as a diagonal mask retaining the first K modes, including the constant mode. The final executable continuous action is

$$A_c^{\text{exec}} = P_K \widehat{A}_c, \qquad P_K = C^\top M_K C. \tag{9}$$

Since P_K is an orthogonal projector, this operation returns the closest trajectory to $\widehat{A}_c$ within the retained temporal subspace. The transform is applied independently to each joint along the action horizon, while the gripper remains unchanged.

When does truncation help? Let A_c^{T} denote the multi-step teacher action under the same conditioning and initial noise, and define mismatch error $E_c = \widehat{A}_c - A_c^{\mathrm{T}}$. To quantify this trade-off, let $\mathcal{R}_K$ denote the overall expected residual error between the projected prediction and the teacher's

action. By the orthogonality of the projection, $\mathcal{R}_K$ explicitly decomposes into two components:

$$\begin{aligned}\mathcal{R}_K &:= \mathbb{E}\left\|P_K\widehat{A}_c - A_c^{\mathrm{T}}\right\|_F^2 \\ &= \underbrace{\mathbb{E}\|P_K E_c\|_F^2}_{\text{retained-band prediction error}} + \underbrace{\mathbb{E}\|(I-P_K)A_c^{\mathrm{T}}\|_F^2}_{\text{discarded teacher energy}}.\end{aligned} \tag{10}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, with $\|X\|_F^2 = \sum_{t=1}^{T}\sum_{j=1}^{d_c} X_{t,j}^2$, and the expectation is taken over conditioning inputs and initial noise.

Relative to the unprojected action, whose error is $\mathcal{R}_{\text{full}} = \mathbb{E}\|E_c\|_F^2$, we have

$$\mathcal{R}_K - \mathcal{R}_{\text{full}} = \mathbb{E}\|(I-P_K)A_c^{\mathrm{T}}\|_F^2 - \mathbb{E}\|(I-P_K)E_c\|_F^2. \tag{11}$$

Truncation therefore improves teacher approximation (i.e., $\mathcal{R}_K < \mathcal{R}_{\text{full}}$) when the discarded frequencies contain more prediction error than teacher motion. Low-frequency concentration of teacher actions alone does not guarantee this condition.

We train the calibrator with the full-band teacher-matching objective described above and apply the fixed projection at inference. The cutoff K controls the trade-off between preserving teacher motion and further suppressing prediction error, $K = T$ recovers the unprojected prediction.

4.3 Joint Execution of Video Prefill and Action Denoising

One-step calibration reduces action generation to a single ActionDiT forward pass. Under the conventional sequential schedule, however, this pass still waits for the entire video prefill to finish. During inference, the video branch produces visual key/value (K/V) features at each layer, which are cached and used to condition the corresponding action block. We exploit this layer-wise conditioning to start action computation as soon as the required visual features become available.

Layer-level dependencies. Starting action computation before video prefill finishes requires identifying the inputs needed by each action block. Let $\mathcal{C}_l$ denote the visual K/V cache produced by video layer l, and let H_{l-1}^a denote the input hidden states of action block l. The action block computes

$$H_l^a = \mathcal{A}_l\left(H_{l-1}^a, \mathcal{C}_l\right), \tag{12}$$

where $\mathcal{A}_l$ denotes the corresponding ActionDiT block. Its execution requires both H_{l-1}^a and $\mathcal{C}_l$, but does not depend on the caches produced by later video layers. Thus, completing the entire video prefill is not a prerequisite for starting action computation.

Joint scheduling. The layer-level dependencies above permit a pipelined execution of video prefill and action generation. We assign the video and action branches to separate CUDA streams and use CUDA events to synchronize access to the visual K/V caches. After video layer l produces $\mathcal{C}_l$, it records an event indicating that the cache is ready. The action stream waits for this event before executing action block l, while the video stream proceeds to the next layer. Execution within the action stream preserves the ordering of action blocks. After the final action block, the calibration head and temporal projection produce the executable action chunk. This schedule preserves the original computational dependencies while allowing video and action computation to overlap.

5 Experiments

We evaluate whether 1Step-WAM preserves task success while reducing inference latency. The latency is measured on a single NVIDIA L40 GPU.

5.1 Experimental Setup

Benchmarks. We evaluate on two simulation benchmarks and a real-robot setup. RoboTwin (Chen et al., 2025) is a bimanual manipulation benchmark with clean and randomized for 50 tasks. LIBERO (Liu et al., 2023) provides complementary evaluations through four task suites: LIBERO-Spatial, LIBERO-Object,LIBERO-Goal, and LIBERO-Long, each containing 10 tasks. For these two

Table 1: Evaluation result on RoboTwin (Chen et al., 2025) simulation benchmark

Model	Method	Train params.	Clean	Random	Avg.	Latency (ms)↓	Speedup ↑
FastWAM (Yuan et al., 2026)	Original	–	91.88	91.78	91.83	669.54	1.00×
	Naive 1-Step	–	69.92	67.52	68.72	147.62	4.54×
	DMD2 (Yin et al., 2024a)	2.04B	82.48	83.40	82.94	147.62	4.54×
	SnapFlow Luan et al. (2026)	1.02B	90.52	86.96	88.74	147.69	4.53×
	Ours (1Step-WAM)	**13.42M**	**91.10**	**90.68**	**90.89**	**89.95**	**7.44×**
ImageWAM (Zhang et al., 2026)	Original	–	93.38	93.30	93.34	306.85	1.00×
	Naive 1-Step	–	37.20	34.53	35.87	77.64	3.95×
	DMD2 (Yin et al., 2024a)	1.28B	64.52	65.40	64.96	77.64	3.95×
	SnapFlow Luan et al. (2026)	643.34M	88.00	88.24	88.12	77.69	3.95×
	Ours (1Step-WAM)	**13.40M**	**92.26**	**92.62**	**92.44**	**51.28**	**5.98×**
Kairos (Team et al., 2026)	Original	–	96.96	94.96	95.96	1831.62	1.000×
	Naive 1-Step	–	55.96	61.00	58.48	133.81	13.69×
	DMD2 (Yin et al., 2024a)	1.90B	70.12	72.00	71.06	133.81	13.69×
	SnapFlow Luan et al. (2026)	0.96B	85.08	82.28	83.68	133.88	13.68×
	Ours (1Step-WAM)	**13.42M**	**96.56**	**93.84**	**95.20**	**73.73**	**24.84×**

task we following the evalation settings in FastWAM (Yuan et al., 2026), ImageWAM (Zhang et al., 2026) and Kairos (Team et al., 2026) For real-world evaluation, we deploy FastWAM-based policies on an AIRBOT Play robotic arm, using a fisheye camera for visual observations. We evaluate three manipulation tasks: (1) *Pick Place*: Pick up all the fruits and vegetables on the table and place them in the basket, (2) *Stack Cups*: stack all the cups on the table, and (3) *Hang Mug*: hang all the mugs on the table. The training set contains 200 episodes per task, and we train the policies for five epochs. During evaluation each task is conducted 20 independent rollouts per task per method and report task-level and average success rates.

Models and Baselines. We apply 1Step-WAM to FastWAM (Yuan et al., 2026), ImageWAM (Zhang et al., 2026), and Kairos (Team et al., 2026). On RoboTwin, we additionally compare with DMD2 (Yin et al., 2024a) distillation model used in video DiTs and SnapFlow (Luan et al., 2026), an action-distillation method developed for VLAs. We adapt these methods to the WAM action branch and evaluate them with the corresponding visual conditioning. Within each backbone, comparisons use the same observation interface, action representation, and action horizon.

Implementation Details. We freeze the pretrained WAM backbone and train only the velocity calibration head for 500 steps using the AdamW optimizer, with a learning rate of 5×10^{-6} and a global batch size of 512. The K used for FastWAM (Yuan et al., 2026), ImageWAM (Zhang et al., 2026) and Kairos Team et al. (2026) is 8,4,8.

5.2 Main Results

Simulation Benchmarks. Table 1 reports success rates and inference latency on RoboTwin (Chen et al., 2025). Across all three backbones, 1Step-WAM achieves the highest success rate among the evaluated one-step methods in both settings. Its average success rates are 90.89%, 92.44%, and 95.20% on FastWAM, ImageWAM, and Kairos, respectively, only 0.94, 0.90, and 0.76 percentage points below their multi-step counterparts. At the same time, end-to-end inference is accelerated by $7.44\times$, $5.98\times$, and $24.84\times$, respectively. Despite outperforming both DMD2 (Yin et al., 2024a) and SnapFlow (Luan et al., 2026) in RoboTwin success rate across all three backbones, 1Step-WAM trains only 13.4M parameters, using fewer trainable parameters than either distillation baseline.

Table 2 reports success rates across the four LIBERO suites. 1Step-WAM closely matches its multi-step counterpart on FastWAM and ImageWAM, with mean success rates of 96.6% versus 97.4% and 98.2% versus 98.4%, respectively. On Kairos, the one-step model reaches 97.2%, exceeding the multi-step baseline by 1.0 percentage point. Compared with naive one-step generation, calibration improves mean success by 0.9, 1.4, and 2.0 percentage points on the three backbones, respectively.

Real-World Task Evaluation. We evaluate FastWAM, naive one-step generation, and 1Step-WAM on an AIRBOT Play manipulator equipped with a fisheye camera. Table 3 reports their success rates.

Table 2: Success rates (%) on the LIBERO suites.

Model	Method	Spatial	Object	Goal	Long	Mean
FastWAM (Yuan et al., 2026)	Original	97.8	99.8	96.0	96.0	97.4
	Naive 1-Step	97.0	**99.2**	95.0	91.6	95.7
	Ours (1Step-WAM)	**97.4**	**99.2**	**95.4**	**94.2**	**96.6**
ImageWAM (Zhang et al., 2026)	Original	97.2	99.2	98.8	98.4	98.4
	Naive 1-Step	95.8	99.6	97.2	94.6	96.8
	Ours (1Step-WAM)	**97.6**	**100.0**	**98.2**	**96.8**	**98.2**
Kairos (Team et al., 2026)	Original	97.4	99.2	97.0	91.2	96.2
	Naive 1-Step	97.6	**100.0**	94.2	89.0	95.2
	Ours (1Step-WAM)	**99.0**	99.0	**98.6**	**92.2**	**97.2**

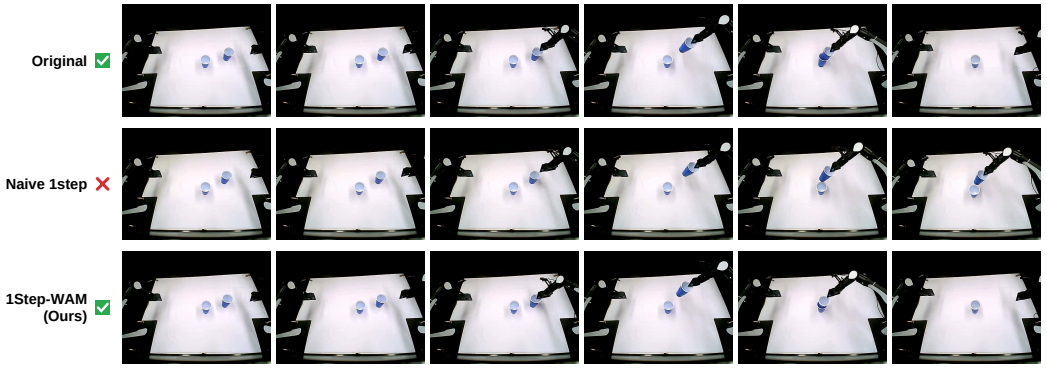


Figure 3: Real-world execution comparison on Stack Cups. In the illustrated rollouts, original FastWAM and 1Step-WAM successfully stack the cups, while naive 1-step fails.

Pick Place tolerates relatively coarse motions, allowing even naive one-step generation to complete the task. Stack Cups and Hang Mug, in contrast, require more precise alignment, making them more sensitive to action prediction errors. Figure 3 compares representative execution sequences of the three methods on Stack Cups. In the illustrated rollouts, both original FastWAM and 1Step-WAM successfully stack the cups, whereas naive one-step generation fails. Additional execution comparisons on Pick Place and Hang Mug are provided in the appendix.

5.3 Empirical Analysis of Denoising Trajectories

We examine whether the denoising trace of FastWAM, ImageWAM, and Kairos are consistent with the analysis in Section 3. Figure 4 reports the cosine similarity between predicted velocities at adjacent denoising steps and their ℓ_2 magnitudes for a representative action chunk. The adjacent step denoise velocity is extremely similar across denoising step, demonstrating the denoise trace is straight. Corresponding results for ImageWAM and Kairos are provided in the appendix.

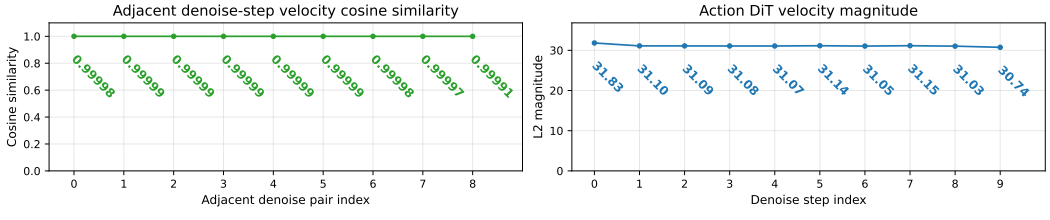


Figure 4: Left: cosine similarity between velocities predicted at adjacent denoising steps. Right: the ℓ_2 magnitude of the predicted velocity at each step.

Table 3: Real World Tasks success rates (%)

Method	Pick Place	Stack Cups	Hang Mug	Avg.
Original FastWAM	100	90	50	80
Naive 1-Step	100	40	10	50
1Step-WAM	**100**	**80**	**45**	**75**

Table 4: Ablation of bandwidth-constrained action decoding on RoboTwin after retaining first K modes

Model	Method	K	Clean	Randomized
FastWAM	Ours	8	$89.6 \rightarrow \mathbf{91.1}$	$88.7 \rightarrow \mathbf{90.7}$
	DMD2	8	$82.5 \rightarrow \mathbf{84.1}$	$83.4 \rightarrow \mathbf{83.6}$
	SnapFlow	8	$90.5 \rightarrow \mathbf{91.3}$	$87.0 \rightarrow \mathbf{91.4}$
ImageWAM	Ours	4	$91.6 \rightarrow \mathbf{92.3}$	$91.7 \rightarrow \mathbf{92.6}$
Kairos	Ours	8	$94.8 \rightarrow \mathbf{96.6}$	$92.2 \rightarrow \mathbf{93.8}$

Table 5: Ablation of joint video–action execution. "w/o joint execution" denotes sequential execution of video prefill and one-step action generation, whereas "w/ joint execution" uses the proposed joint execution schedule.

Model	w/o Joint (ms) ↓	w/ Joint (ms) ↓	Speedup ↑	Reduction (%) ↑
FastWAM (Yuan et al., 2026)	149.05	89.95	1.66×	39.65
ImageWAM (Zhang et al., 2026)	77.64	51.28	1.51×	33.95
Kairos (Team et al., 2026)	134.78	73.73	1.83×	45.30

5.4 Ablation Studies

Ablation of Bandwidth-Constrained Action Decoding. We evaluate the contribution of bandwidth-constrained action decoding by comparing the same calibrated checkpoint with and without it. Table 4 reports success before and after projection and the change in percentage points, it improves 1Step-WAM success rates under both clean and randomized settings across all three backbones, with gains ranging from 0.7 to 2.0 percentage points, demonstrating the effectiveness of bandwidth-constrained action decoding. We also apply the same projection to DMD2 and SnapFlow on FastWAM, both methods are improved, suggesting that the benefits extend beyond our approach to other distillation methods.

Analysis of joint execution. Table 5 isolates the effect of jointly scheduling video prefill and one-step action generation. Compared with sequential execution, the proposed schedule reduces end-to-end latency by 39.65%, 33.95%, and 45.30% on FastWAM, ImageWAM, and Kairos, respectively, corresponding to $1.66\times$, $1.51\times$, and $1.83\times$ speedups.

6 Conclusion

We propose 1Step-WAM, a lightweight framework for converting pretrained multi-step World Action Models into one-step action generators. Motivated by the concentration of conditional action distributions and the approximately straight denoising trajectories, we introduced a small calibration head that corrects the initial velocity prediction while keeping the backbone frozen. We further improved action quality through bandwidth-constrained decoding and propose a joint execution schedule to overlap visual prefill with action generation. Evaluations on RoboTwin, LIBERO, and real-world manipulation tasks show that 1Step-WAM largely preserves multi-step task success with a single action-generation pass and with considerable speedup.

AI USE STATEMENT

In this work, we utilized generative AI tools to assist with drafting, editing, and polishing text, as well as generating routine code for data analysis and plotting. The authors have meticulously reviewed and revised all AI-assisted text, and thoroughly tested all AI-generated code. We confirm that every reported result and metric originates entirely from our own physical and simulated experiments. The authors assume full responsibility for the final content, claims, and artifacts presented in this paper.

REFERENCES

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

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.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2025.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Black Forest Labs. Flux. 2: Frontier visual intelligence. *https://bfl. ai/blog/flux-2*, 2025.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Galaxea Team. Galaxea g0: Open-world dataset and dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Kairos Team, Fei Wang, Shan You, Qiming Zhang, Tao Huang, Zuoyi Fu, Zhisheng Zheng, Yunlong Xi, Feng Lv, Xiaoming Wu, Zeyu Liu, Cong Wan, Pu Li, Ruiqing Yang, Xiaoou Li, Wei Wang, Kangkang Zhu, Yuwei Zhang, Shi Fu, Zheng Zhang, Xiaoning Wu, Xuzeng Fan, Dacheng Tao, and Xiaogang Wang. Kairos: A regret-aware native world-action model stack for physical ai, 2026. URL https://arxiv.org/abs/2606.16533.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026.

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

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2024b.

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

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. ImageWAM: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026.

7 APPENDIX

7.1 DETAILED DERIVATION

We provide the detailed velocity derivation underlying Section 3.

7.1.1 THEORETICAL SETUP

Distributions and independent coupling. Let d denote the action dimension. For fixed conditioning inputs, we assume a Gaussian clean target action distribution and a Gaussian noise prior:

$$x_0 \sim \mathcal{N}(\mu, \sigma^2 I_d), \qquad x_1 \sim \mathcal{N}(0, I_d), \qquad x_0 \perp x_1, \tag{13}$$

where $\mu \in \mathbb{R}^d$ is the target action center and $0 < \sigma^2 \ll 1$ is the residual variance. The independence assumption applies to the endpoints sampled for the training interpolation.

Linear interpolation and marginal distribution. The conditional flow-matching interpolation defines the forward noising path:

$$x_t = (1-t)x_0 + tx_1, \qquad t \in [0,1]. \tag{14}$$

Denoising traverses time in the reverse direction, from $t = 1$ to $t = 0$. By independence and the linearity of Gaussian random variables,

$$\begin{aligned} \mathbb{E}[x_t] &= (1-t)\mathbb{E}[x_0] + t\mathbb{E}[x_1] = (1-t)\mu, \\ \mathrm{Var}(x_t) &= (1-t)^2 \,\mathrm{Var}(x_0) + t^2 \,\mathrm{Var}(x_1) \\ &= \big[(1-t)^2\sigma^2 + t^2\big] I_d. \end{aligned} \tag{15}$$

Define the time-dependent variance and its derivative as

$$\begin{aligned} D(t) &= (1-t)^2\sigma^2 + t^2, \\ D'(t) &= -2(1-t)\sigma^2 + 2t = 2\big[t - (1-t)\sigma^2\big]. \end{aligned} \tag{16}$$

The marginal distribution is therefore

$$x_t \sim \mathcal{N}\big((1-t)\mu, D(t) I_d\big). \tag{17}$$

7.1.2 DERIVATION OF THE OPTIMAL VELOCITY FIELD

The optimal flow-matching velocity field is defined with respect to increasing t as the conditional expectation of the interpolation velocity:

$$v^*(x_t, t) = \mathbb{E}[x_1 - x_0 \mid x_t]. \tag{18}$$

For jointly Gaussian random vectors A and B, the conditional expectation identity is

$$\mathbb{E}[A \mid B] = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(B - \mu_B). \tag{19}$$

We apply this identity with $A = x_1 - x_0$ and $B = x_t$.

Means and covariances. The means and the covariance of B are

$$\begin{aligned} \mu_A &= \mathbb{E}[x_1 - x_0] = -\mu, \\ \mu_B &= \mathbb{E}[x_t] = (1-t)\mu, \\ \Sigma_{BB} &= \mathrm{Var}(x_t) = D(t) I_d. \end{aligned} \tag{20}$$

Using the bilinearity of covariance and $x_0 \perp x_1$, the cross-covariance is

$$\begin{aligned} \Sigma_{AB} &= \mathrm{Cov}\big(x_1 - x_0, (1-t)x_0 + tx_1\big) \\ &= \mathrm{Cov}(x_1, tx_1) - \mathrm{Cov}(x_0, (1-t)x_0) \\ &= t\,\mathrm{Var}(x_1) - (1-t)\,\mathrm{Var}(x_0) \\ &= \big[t - (1-t)\sigma^2\big] I_d. \end{aligned} \tag{21}$$

Substitution and simplification. Substituting these quantities into the conditional expectation gives

$$\begin{aligned} v^*(x_t,t) &= -\mu + \big[t-(1-t)\sigma^2\big] I_d \big[D(t) I_d\big]^{-1} \big[x_t - (1-t)\mu\big] \\ &= -\mu + \frac{t-(1-t)\sigma^2}{D(t)} \big[x_t - (1-t)\mu\big] \\ &= \frac{\big\{-D(t) - (1-t)[t-(1-t)\sigma^2]\big\}\mu + [t-(1-t)\sigma^2]x_t}{D(t)}. \end{aligned} \tag{22}$$

The numerator of the coefficient of μ simplifies to

$$\begin{aligned} -D(t) - (1-t)\big[t-(1-t)\sigma^2\big] &= -(1-t)^2\sigma^2 - t^2 - t(1-t) + (1-t)^2\sigma^2 \\ &= -t^2 - t(1-t) \\ &= -t\big[t + (1-t)\big] = -t. \end{aligned} \tag{23}$$

We thus obtain the closed-form velocity field:

$$v^*(x_t,t) = -\frac{t}{D(t)}\mu + \frac{t-(1-t)\sigma^2}{D(t)} x_t. \tag{24}$$

7.1.3 Analytical solution of the inference ODE

During inference, we integrate the optimal velocity field backward from $t=1$ to $t=0$, starting from a noise sample x_1. In this subsection, x_t denotes the ODE trajectory, which shares the marginals of the training interpolation but couples its endpoints through the ODE:

$$\dot{x}_t = \frac{t-(1-t)\sigma^2}{D(t)} x_t - \frac{t}{D(t)}\mu, \qquad x(1) = x_1. \tag{25}$$

Here, $\dot{x}_t$ is the derivative with respect to increasing t; denoising updates use $\Delta t < 0$. Writing the equation as $\dot{x}_t + P(t)x_t = Q(t)$ gives

$$P(t) = -\frac{t-(1-t)\sigma^2}{D(t)} = -\frac{D'(t)}{2D(t)}, \qquad Q(t) = -\frac{t}{D(t)}\mu. \tag{26}$$

Integrating factor. The integrating factor is

$$\begin{aligned} I(t) &= \exp\left(\int_1^t P(\tau)\, d\tau\right) \\ &= \exp\left(-\frac{1}{2}\int_1^t \frac{D'(\tau)}{D(\tau)}\, d\tau\right) \\ &= \exp\left(-\frac{1}{2}\ln D(t) + \frac{1}{2}\ln D(1)\right). \end{aligned} \tag{27}$$

Since $D(1)=1$, this reduces to $I(t) = D(t)^{-1/2}$. Multiplying the ODE by this factor yields

$$D(t)^{-1/2}\dot{x}_t - \frac{1}{2}D(t)^{-3/2}D'(t)x_t = -\frac{t}{D(t)^{3/2}}\mu. \tag{28}$$

By the product rule, the left-hand side is the derivative of $D(t)^{-1/2}x_t$. Therefore,

$$\frac{d}{dt}\left[\frac{x_t}{\sqrt{D(t)}}\right] = -\frac{t}{D(t)^{3/2}}\mu. \tag{29}$$

Antiderivative of the right-hand side. Differentiating $(1-t)/\sqrt{D(t)}$ gives

$$\begin{aligned} \frac{d}{dt}\left[\frac{1-t}{\sqrt{D(t)}}\right] &= \frac{-\sqrt{D(t)} - \frac{1}{2}(1-t)D(t)^{-1/2}D'(t)}{D(t)} \\ &= \frac{-D(t) - \frac{1}{2}(1-t)D'(t)}{D(t)^{3/2}}. \end{aligned} \tag{30}$$

Substituting $D(t)$ and $D'(t)$ into the numerator,

$$\begin{aligned} -D(t) - \frac{1}{2}(1-t)D'(t) &= -(1-t)^2\sigma^2 - t^2 - (1-t)\big[t - (1-t)\sigma^2\big] \\ &= -t^2 - t(1-t) = -t. \end{aligned} \tag{31}$$

Hence

$$\frac{d}{dt}\left[\frac{1-t}{\sqrt{D(t)}}\right] = -\frac{t}{D(t)^{3/2}}. \tag{32}$$

Inference trajectory and instantaneous velocity. Integrating Equation (29) from 1 to t gives

$$\int_1^t \frac{d}{d\tau}\left[\frac{x_\tau}{\sqrt{D(\tau)}}\right] d\tau = -\int_1^t \frac{\tau}{D(\tau)^{3/2}}\, d\tau \cdot \mu, \tag{33}$$

and therefore

$$\frac{x_t}{\sqrt{D(t)}} - \frac{x_1}{\sqrt{D(1)}} = \left[\frac{1-\tau}{\sqrt{D(\tau)}}\right]_1^t \mu. \tag{34}$$

Using $D(1) = 1$ and multiplying by $\sqrt{D(t)}$, we obtain

$$x_t = \sqrt{D(t)}\, x_1 + (1-t)\mu = \sqrt{(1-t)^2\sigma^2 + t^2}\, x_1 + (1-t)\mu. \tag{35}$$

At $t = 0$, the endpoint is $x_{t=0} = \mu + \sigma x_1$, which follows the target distribution $\mathcal{N}(\mu, \sigma^2 I_d)$. Differentiation with respect to t gives the instantaneous velocity:

$$\dot{x}_t = -\mu + C(t)x_1, \qquad C(t) = \frac{D'(t)}{2\sqrt{D(t)}} = \frac{t - (1-t)\sigma^2}{\sqrt{(1-t)^2\sigma^2 + t^2}}. \tag{36}$$

7.1.4 Main-interval collinearity and terminal boundary behavior

Theorem 1. For any fixed $\epsilon \in (0,1)$ independent of σ, as $\sigma \to 0^+$, $C(t)$ converges uniformly to 1 on $[\epsilon, 1]$. For a fixed noise realization x_1, the ODE velocity therefore approaches $x_1 - \mu$ throughout this interval, and the reverse-time denoising velocity $-\dot{x}_t$ approaches $\mu - x_1$. Near the clean endpoint, the coefficient crosses zero at $t^* = \sigma^2/(1+\sigma^2)$, where $\dot{x}_{t^*} = -\mu$.

Main-interval collinearity. For $t \in [\epsilon, 1]$, we have $t \geq \epsilon > 0$ and can write

$$C(t) = \frac{1 - \frac{(1-t)\sigma^2}{t}}{\sqrt{1 + \left(\frac{(1-t)\sigma}{t}\right)^2}}. \tag{37}$$

The two ratios satisfy the uniform bounds

$$0 \leq \frac{(1-t)\sigma}{t} \leq \frac{(1-\epsilon)\sigma}{\epsilon} = \mathcal{O}(\sigma), \qquad 0 \leq \frac{(1-t)\sigma^2}{t} \leq \frac{(1-\epsilon)\sigma^2}{\epsilon} = \mathcal{O}(\sigma^2). \tag{38}$$

The constants in these bounds may depend on ϵ but not on t or σ. Consequently,

$$C(t) = \frac{1 + \mathcal{O}(\sigma^2)}{\sqrt{1 + \mathcal{O}(\sigma^2)}}, \qquad \lim_{\sigma \to 0^+} \sup_{t \in [\epsilon, 1]} |C(t) - 1| = 0. \tag{39}$$

It follows that $\dot{x}_t = -\mu + C(t)x_1$ converges uniformly to $x_1 - \mu$ on this interval. For $x_1 - \mu \neq 0$ and any $t_1, t_2 \in [\epsilon, 1]$, the velocity cosine similarity satisfies

$$\lim_{\sigma \to 0^+} \cos(\dot{x}_{t_1}, \dot{x}_{t_2}) = \frac{(x_1 - \mu)^\top (x_1 - \mu)}{\|x_1 - \mu\|_2 \, \|x_1 - \mu\|_2} = 1. \tag{40}$$

Negating both velocities leaves this cosine similarity unchanged, so the same collinearity holds in the denoising direction.

Terminal transition scale. The main-interval convergence does not extend uniformly to $t = 0$, since $C(0) = -\sigma \to 0$ rather than 1. To resolve the terminal transition, set $t = \alpha\sigma$ for fixed $\alpha \geq 0$. Then

$$C(\alpha\sigma) = \frac{\alpha - (1 - \alpha\sigma)\sigma}{\sqrt{\alpha^2 + (1 - \alpha\sigma)^2}} \longrightarrow \frac{\alpha}{\sqrt{1 + \alpha^2}}. \tag{41}$$

This limiting profile describes the change in the noise coefficient on the scale $t = \mathcal{O}(\sigma)$ near the clean endpoint, consistent with Section 3.

Terminal zero crossing. Solving $C(t^*) = 0$ gives

$$t^* - (1 - t^*)\sigma^2 = 0 \quad \Longrightarrow \quad t^* = \frac{\sigma^2}{1 + \sigma^2} = \sigma^2 + \mathcal{O}(\sigma^4). \tag{42}$$

Thus the zero crossing lies $\mathcal{O}(\sigma^2)$ from the clean endpoint $t = 0$. At this time the noise term in the instantaneous velocity vanishes:

$$C(t^*) = 0, \qquad \dot{x}_{t^*} = -\mu. \tag{43}$$

For $t \in (t^*, 1]$, $C(t) > 0$, whereas for $t \in [0, t^*)$, $C(t) < 0$. At the clean endpoint,

$$C(0) = -\frac{\sigma^2}{\sqrt{\sigma^2}} = -\sigma, \qquad \dot{x}_{t=0} = -\mu - \sigma x_1. \tag{44}$$

The reverse-time denoising velocity therefore approaches μ at the clean endpoint as $\sigma \to 0^+$. The scale $\mathcal{O}(\sigma^2)$ specifies the zero-crossing location within the broader $\mathcal{O}(\sigma)$ terminal transition region.

One-step estimate and residual error. At the noise endpoint, $D(1) = 1$ and $v^*(x_1, 1) = x_1 - \mu$. A single reverse-time Euler step with $\Delta t = -1$ therefore gives

$$\widehat{x}_0 = x_1 - v^*(x_1, 1) = \mu. \tag{45}$$

For the same initial noise, the exact ODE endpoint is $x_{t=0} = \mu + \sigma x_1$. Hence, under this Gaussian model and the optimal velocity field,

$$\widehat{x}_0 - x_{t=0} = -\sigma x_1, \qquad \mathbb{E}_{x_1} \left\| \widehat{x}_0 - x_{t=0} \right\|_2^2 = d\sigma^2. \tag{46}$$

Thus the one-step estimate recovers the conditional mean and approximates the ODE endpoint as $\sigma \to 0^+$ for fixed d; it does not reproduce the residual output variance at finite σ.

7.2 Per-Joint Spectral Analysis

Figure 5 compares the temporal DCT coefficients of FastWAM teacher actions and the residual errors of the calibrated one-step predictions. The transform is applied independently along the action horizon for each continuous joint dimension. The blue curves show the teacher coefficients, while the red curves show the coefficients of the teacher-minus-prediction residual before bandwidth-constrained decoding. Since the plotted coefficients are signed, we compare their absolute magnitudes; the energy of each mode is given by the squared coefficient.

The teacher coefficients are largest in the low-frequency modes and become small at higher frequencies. Residual errors nevertheless remain visible in some of these higher-frequency modes. In particular, joint dimensions 5 and 12 exhibit modes where the error magnitude exceeds that of the corresponding teacher component. Prediction errors can therefore overwhelm weak high-frequency teacher signals, even when these errors are small relative to the dominant low-frequency components.

This observation motivates retaining the first K DCT modes to preserve the dominant teacher components while suppressing residual errors in the discarded frequencies. As established in our projection analysis, truncation reduces teacher-matching error when the total discarded error energy exceeds the total discarded teacher energy.

7.3 Real Tasks Rollout on Pick Place and Hang Mug

The real execution sequences of the three methods on Pick Place and Hang Mug are shown as Figure 6 and Figure 7.

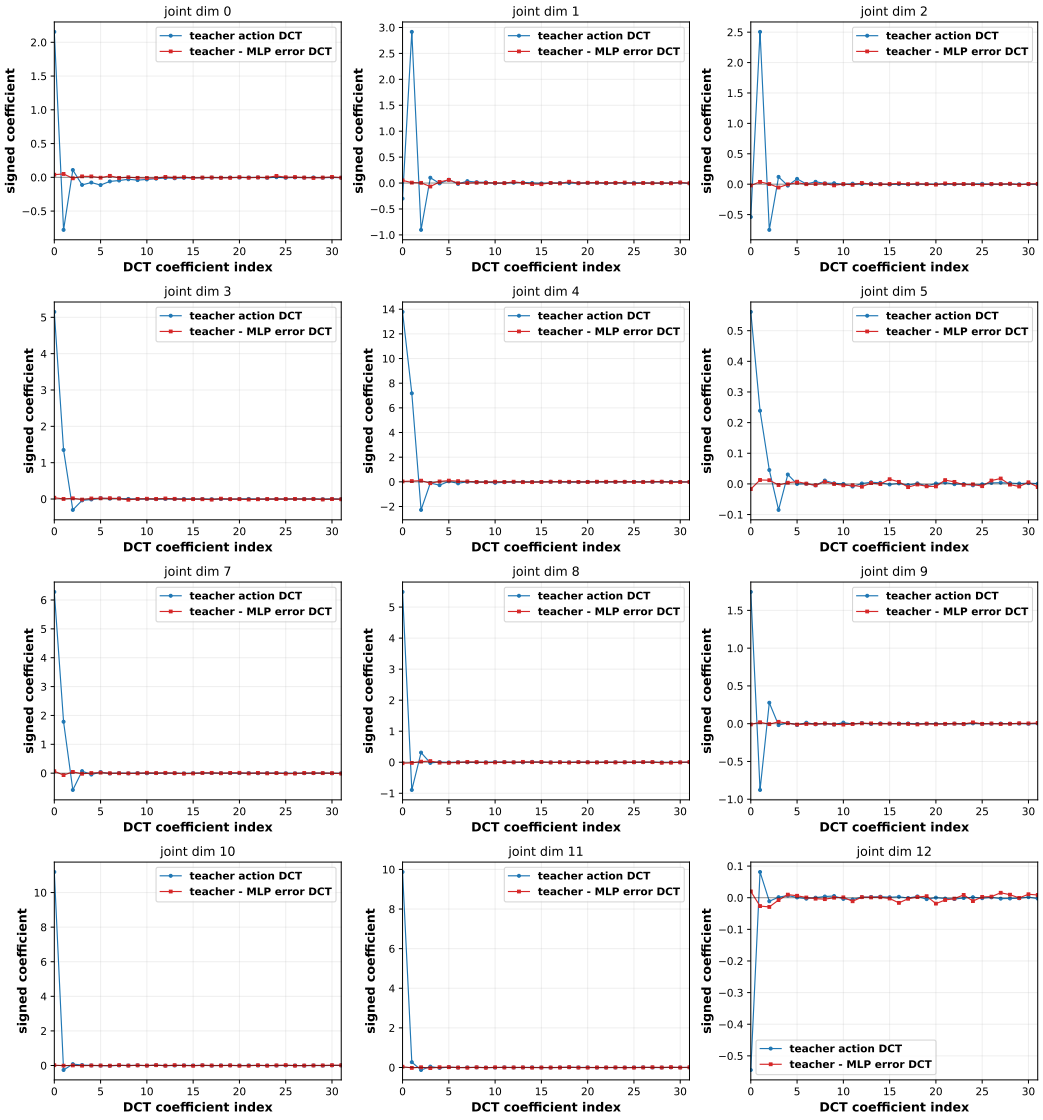


Figure 5: Per-joint temporal DCT coefficients of FastWAM teacher actions (blue) and teacher-minus-prediction residuals from the calibrated one-step model (red). Each panel corresponds to one continuous joint dimension, with frequency increasing along the horizontal axis.

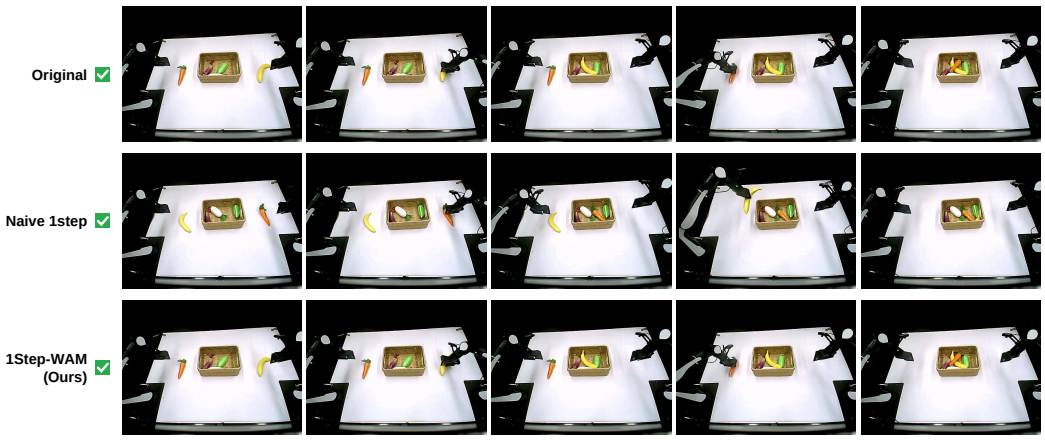


Figure 6: Real-world execution comparison on pick place. In the illustrated rollouts, both are success.

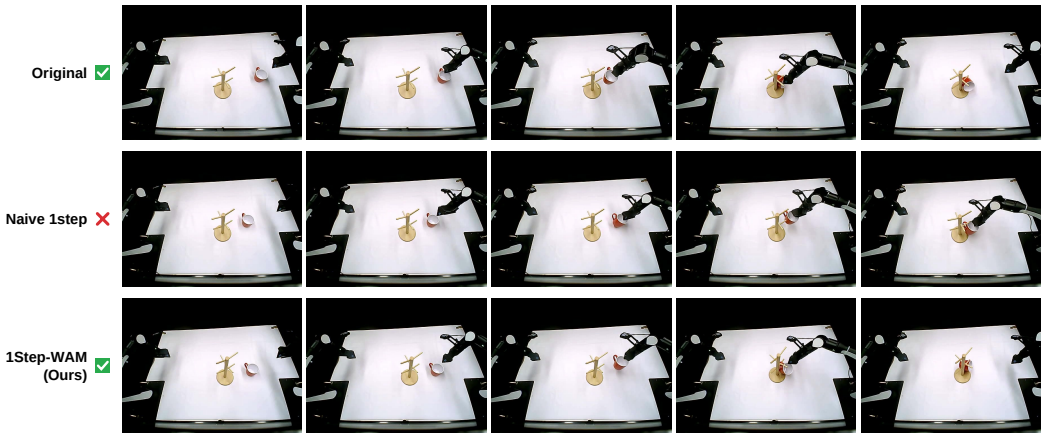


Figure 7: Real-world execution comparison on hang mug. In the illustrated rollouts, original FastWAM and 1Step-WAM successfully hang mug, while naive 1-step fails.

7.4 Empirical Analysis of Denoising Trajectories on ImageWAM and Kairos

Figure 8 and Figure 9 reports the cosine similarity between predicted velocities at adjacent denoising steps and their ℓ_2 magnitudes for a representative action chunk. Both adjacent step denoise velocity is extremely similar across denoising step, demonstrating the denoise trace is straight.

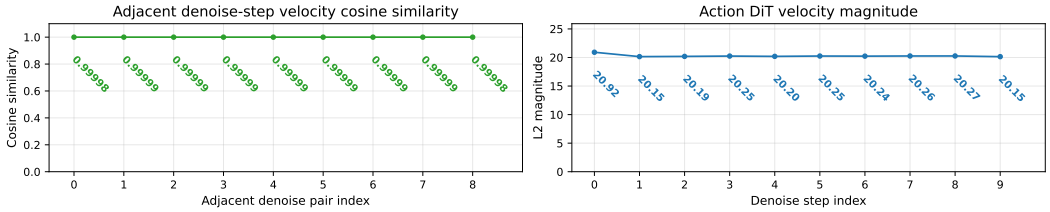


Figure 8: Left: cosine similarity between velocities predicted at adjacent denoising steps. Right: the ℓ_2 magnitude of the predicted velocity at each step.

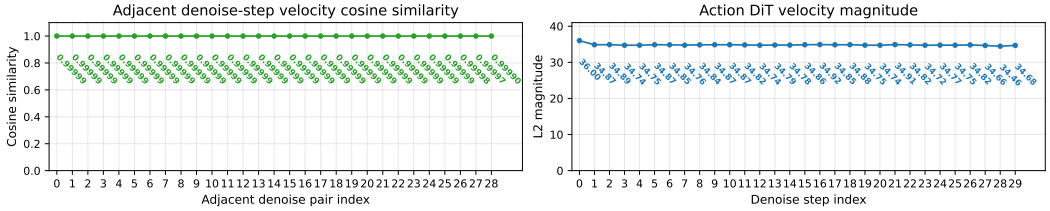


Figure 9: Left: cosine similarity between velocities predicted at adjacent denoising steps. Right: the ℓ_2 magnitude of the predicted velocity at each step.

7.5 Detailed RoboTwin result

Per-task detailed result of our proposed 1Step-WAM is shown as Table 6

Table 6: Comparison of Fast-WAM, ImageWAM, and Kairos across manipulation tasks.

Task	Fast-WAM				ImageWAM				Kairos			
	Clean		Rand.		Clean		Rand.		Clean		Rand.	
	Original	Ours	Original	Ours	10-steps	Ours	10-steps	Ours	Original	Ours	Original	Ours
adjust_bottle	1.00	1.0	1.00	0.99	1.00	1.0	0.99	0.99	1.0	1.0	1.0	1.0
beat_block_hammer	0.99	1.0	0.97	0.98	1.00	0.98	0.99	0.9	1.0	1.0	0.98	1.0
blocks_ranking_rgb	1.00	1.0	1.00	0.98	0.99	0.9	0.99	0.98	1.0	1.0	1.0	1.0
blocks_ranking_size	0.94	0.91	0.98	0.92	0.95	0.9	0.98	0.96	1.0	1.0	0.92	0.96
click_alarmclock	1.00	1.0	1.00	1.0	1.00	0.99	1.00	1.0	1.0	1.0	1.0	1.0
click_bell	1.00	1.0	1.00	1.0	1.00	1.0	1.00	1.0	1.0	1.0	1.0	1.0
dump_bin_bigbin	0.97	0.93	0.96	0.96	0.95	0.99	0.92	0.95	0.92	0.98	0.96	0.98
grab_roller	1.00	1.0	1.00	1.0	1.00	1.0	1.00	1.0	1.0	1.0	1.0	1.0
handover_block	0.95	0.86	0.81	0.78	0.95	0.97	0.95	0.93	1.0	0.98	0.96	0.92
handover_mic	0.99	0.95	1.00	0.97	0.99	1.0	0.99	1.0	0.98	0.96	0.96	1.0
hanging_mug	0.58	0.66	0.62	0.56	0.77	0.64	0.73	0.62	0.98	0.96	0.62	0.56
lift_pot	1.00	0.99	1.00	1.0	1.00	0.99	1.00	1.0	1.0	1.0	1.0	1.0
move_can_pot	0.90	0.93	0.88	0.92	0.95	0.88	0.97	0.87	0.98	0.98	0.98	0.92
move_pillbottle_pad	1.00	0.98	0.99	0.98	0.99	0.98	1.00	1.0	1.0	1.0	1.0	0.98
move_playingcard_away	1.00	1.0	1.00	1.0	1.00	1.0	0.99	0.99	1.0	0.98	1.0	1.0
move_stapler_pad	0.77	0.72	0.64	0.67	0.63	0.62	0.70	0.67	0.94	0.9	0.94	0.9
open_laptop	0.98	0.96	1.00	0.99	0.98	0.97	0.99	0.98	1.0	0.88	1.0	0.92
open_microwave	0.62	0.74	0.45	0.58	0.96	0.98	0.96	0.99	0.88	0.96	0.98	1.0
pick_diverse_bottles	0.80	0.88	0.85	0.88	0.85	0.83	0.89	0.89	0.96	0.94	0.94	0.94
pick_dual_bottles	1.00	0.99	0.96	0.96	0.99	0.91	0.99	0.92	1.0	1.0	1.0	0.96
place_a2b_left	0.95	0.94	0.93	0.95	0.95	0.94	0.96	0.96	1.0	1.0	0.96	0.92
place_a2b_right	0.93	0.95	0.99	0.97	0.98	0.97	0.96	0.96	1.0	1.0	1.0	1.0
place_bread_basket	0.91	0.94	0.93	0.93	0.96	0.96	0.91	0.91	1.0	0.96	0.96	0.96
place_bread_skillet	0.90	0.88	0.93	0.91	0.87	0.91	0.89	0.88	0.92	0.92	0.94	0.82
place_burger_fries	0.96	0.97	0.99	0.97	0.98	0.97	1.00	1.0	0.96	0.98	0.98	0.98
place_can_basket	0.71	0.66	0.69	0.69	0.74	0.79	0.66	0.65	0.86	0.84	0.64	0.56
place_cans_plasticbox	0.99	0.97	0.96	0.97	0.99	1.0	0.96	0.99	0.98	1.0	0.98	0.98
place_container_plate	0.96	1.0	1.00	1.0	0.98	0.96	0.98	1.0	1.0	1.0	1.0	1.0
place_dual_shoes	0.94	0.84	0.88	0.87	0.81	0.82	0.85	0.88	0.9	0.96	0.92	0.9
place_empty_cup	1.00	0.99	1.00	1.0	1.00	1.0	1.00	1.0	1.0	1.0	1.0	1.0
place_fan	0.96	0.97	0.96	0.89	0.97	0.94	0.94	0.94	1.0	1.0	0.96	0.92
place_mouse_pad	0.83	0.74	0.89	0.81	0.86	0.88	0.90	0.87	1.0	0.98	1.0	1.0
place_object_basket	0.89	0.81	0.88	0.81	0.74	0.85	0.77	0.79	0.86	0.88	0.88	0.94
place_object_scale	0.90	0.92	0.97	0.95	0.96	0.96	0.96	0.96	1.0	0.98	0.96	0.98
place_object_stand	0.90	0.94	0.94	0.95	0.98	0.96	0.95	0.97	0.98	0.98	0.96	0.92
place_phone_stand	0.97	0.98	0.99	0.98	1.00	0.99	1.00	0.97	1.0	1.0	1.0	0.96
place_shoe	0.96	0.97	0.99	0.99	0.97	0.97	0.97	0.98	1.0	1.0	0.98	1.0
press_stapler	0.90	0.91	0.97	0.94	0.97	0.95	1.00	1.0	1.0	1.0	0.94	0.94
put_bottles_dustbin	0.95	0.91	0.90	0.82	0.97	0.97	0.90	0.92	0.96	0.94	0.92	0.98
put_object_cabinet	0.94	0.82	0.89	0.85	0.91	0.92	0.91	0.92	0.84	0.88	0.9	0.9
rotate_qrcode	0.93	0.86	0.89	0.89	0.88	0.88	0.92	0.9	0.9	0.84	0.96	0.94
scan_object	0.89	0.84	0.92	0.84	0.95	0.89	0.89	0.94	0.94	1.0	0.94	0.92
shake_bottle	1.00	1.0	1.00	1.0	1.00	1.0	1.00	1.0	1.0	1.0	1.0	1.0
shake_bottle_horizontally	1.00	1.0	1.00	1.0	1.00	1.0	1.00	1.0	1.0	1.0	1.0	1.0
stack_blocks_three	0.95	0.93	0.97	0.96	0.96	0.92	0.99	0.96	0.98	0.98	0.96	0.94
stack_blocks_two	1.00	1.0	1.00	1.0	1.00	0.98	0.99	0.98	1.0	1.0	1.0	1.0
stack_bowls_three	0.80	0.84	0.81	0.83	0.80	0.8	0.79	0.82	0.82	0.78	0.84	0.9
stack_bowls_two	0.92	0.96	0.98	0.95	0.97	0.95	0.96	0.97	0.98	1.0	0.98	1.0
stamp_seal	0.90	0.89	0.94	0.88	0.83	0.73	0.83	0.81	1.0	0.98	0.88	0.82
turn_switch	0.61	0.62	0.59	0.62	0.76	0.74	0.78	0.74	0.96	0.88	0.8	0.7
Mean	0.9188	0.9110	0.9178	0.9068	0.9338	0.9226	0.9330	0.9262	0.9696	0.9656	0.9496	0.9384