# ANYSTEP-WAM: BUDGET-ALIGNED DISTILLATION AND ADAPTIVE INFERENCE FOR WORLD ACTION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

World-action models (WAMs) couple predictive visual modeling with action generation, typically relying on iterative denoising with a fixed denoising steps. However, manipulation tasks contain actions chunks with varying sensitivity to generation errors: critical actions require precision, while less sensitive actions allow faster generation with fewer denoising steps. Here we introduce **AnyStep World Action Model**, a general framework for tunable-budget prediction and scene-dependent computation allocation. Our budget-aligned teacher-trajectory distillation trains interval-conditioned flow maps using explicit frozen-teacher transitions and shared low-rank adapters, supporting action generation from one-step prediction to multi-step refinement. Building on this capability, a lightweight risk-benefit scheduler predicts teacher-curvature-based difficulty and budget-specific student-teacher fidelity from a single one-step preview, selecting the smallest budget predicted to satisfy risk-adaptive fidelity requirements. We evaluate our framework on three widely used WAMs Motus, FastWAM, and LingBotVA using RoboTwin 2.0. Our method reduces average denoising steps by 60.2%, 49.8%, and 85.28%, respectively, while maintaining baseline task success rates. In particular, our AnyStep training substantially improves model performance under a one-step denoising budget, increasing task success rates by 7.07%, 12.08%, and 8.94% on Motus, FastWAM, and LingBotVA, respectively. Experiments on six real-world manipulation tasks further validate its effectiveness.

## 1 INTRODUCTION

World action models (WAMs) (Bi et al., 2026; Li et al., 2026b; Yuan et al., 2026) couple predictive visual modeling with action generation and have demonstrated strong performance in robotic manipulation. However, their repeated denoising through large diffusion transformers (DiTs) delays responses to new observations. Reducing this iterative computation while preserving manipulation performance is therefore critical for efficient deployment.

To reduce this cost, consistency learning (Luo et al., 2023) and distribution matching distillation (Yin et al., 2024b;a) enable one- or few-step image and video generation. Flash-WAM (Akbari et al., 2026) extends consistency distillation to WAMs, accounting for distinct video and action noise regimes to enable one-step prediction. However, fixed budgets cannot accommodate varying precision demands: some motions may need little refinement, while contact-sensitive actions may require more (Figure 1 (a)). AnyFlow (Gu et al., 2026) supports variable-step video generation, but its finite-difference supervision depends on the evolving student (Figure 1 (b)). In our WAM experiments, adapting AnyFlow produces highly variable targets and frequent gradient spikes during training, alongside low task success at one or two steps. The first challenge is therefore to train a single WAM that accurately predicts finite-interval action transitions across denoising budgets.

Accurate prediction across budgets leaves a separate question: *how many* steps should the WAM use for the current action chunk? Adaptive methods in diffusion generation (Zhang et al., 2025; Le et al., 2026) and robotic policies (Hu et al., 2024; Han et al., 2026; Ang et al., 2026; Li et al., 2026a) vary computation with the input. For a WAM, changes in the teacher's denoising trajectory can indicate when coarse updates may be difficult, but they do not reveal how accurately the distilled student



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.

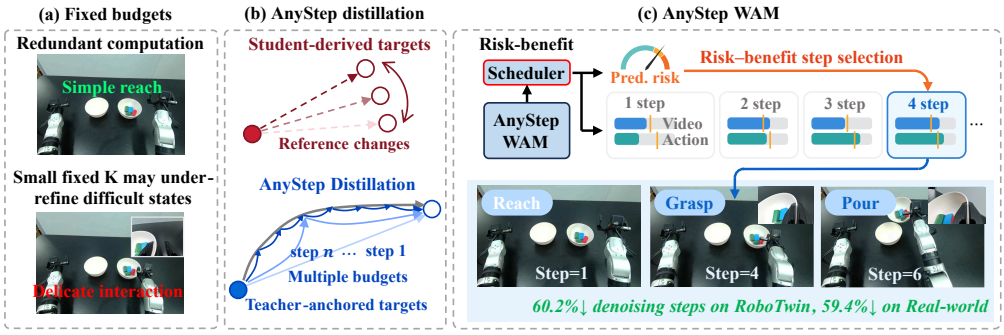


Figure 1: Overview of AnyStep WAM. (a) Fixed budgets can waste computation on easy states or under-refine difficult ones. (b) Teacher-anchored interval supervision enables prediction across budgets. (c) Risk-benefit scheduling adapts denoising steps to scene difficulty and predicted fidelity.

predicts at a particular budget. Step selection must therefore consider both denoising difficulty and the student's budget-dependent fidelity. The second challenge is to estimate these quantities online and choose the smallest budget expected to provide sufficient fidelity.

As illustrated in Figure 1 (c), we introduce **Any-Step World Action Model (AnyStep WAM)** for prediction across denoising budgets and adaptive inference. Its central training idea is *budget-aligned integral flow-map distillation*: we sample intervals from candidate inference schedules and supervise each finite transition with the corresponding displacement along a frozen teacher trajectory. These targets supervise both long one-step and shorter multi-step updates without estimating temporal derivatives of the evolving student. We also train corrective continuation from one-step student preview states, enabling preview reuse and refinement after budget selection. Shared low-rank adapters support all candidate budgets within a single adapted WAM.

Building on this capability, we formulate adaptive inference as a **risk-benefit decision problem**. Teacher-trajectory variation provides a difficulty proxy, while budget-specific student–teacher agreement estimates attainable fidelity. From a single one-step preview, a lightweight scheduler predicts both quantities and selects the smallest budget whose predicted fidelity meets a difficulty-dependent threshold. Reusing the preview in the selected trajectory avoids extra denoising passes, online teacher evaluation, and candidate rollouts.

We evaluate AnyStep WAM on Motus (Bi et al., 2026), LingBotVA (Li et al., 2026b), and FastWAM (Yuan et al., 2026) using RoboTwin (Chen et al., 2025) and six real-world tasks. On RoboTwin, distillation improves one-step success by 7.07, 8.94, and 12.08 percentage points on Motus, LingBotVA, and FastWAM, respectively. Adaptive inference reduces their mean denoising steps by 60.2%, 85.28%, and 49.8%, respectively, while keeping average success within 0.24 percentage points of the corresponding full-budget baselines. On the real robot, it achieves 1.67–6.14$\times$ per-call inference speedups with comparable or higher average success rates across the six tasks.

Our contributions are fourfold. **(i) Budget-aligned WAM distillation:** frozen-teacher interval targets and continuation training from student-generated states enable accurate prediction across multiple denoising budgets within one model. **(ii) Risk-benefit adaptive inference:** We estimate teacher-derived denoising difficulty and the student's fidelity at each budget, then adapt the step count to each interaction by selecting the smallest budget predicted to meet its fidelity requirement. **(iii) Efficient single-preview execution:** a one-step preview informs budget selection and is reused in the trajectory, avoiding an extra denoising pass, online teacher access, and candidate rollouts. **(iv) Experimental validation:** evaluations of three WAM backbones on RoboTwin and six real-world tasks show substantial reductions in denoising steps and latency while maintaining comparable or higher task success.

## 2 Related Work

**Few- and any-step generation.** Consistency learning (Luo et al., 2023) and distribution matching distillation (Yin et al., 2024b;a) compress iterative diffusion or flow generation into one or a few steps, with related advances in robotic policies (Yan et al., 2025; Chen et al., 2026; Gao et al.,

2026) and joint video-action generation (Akbari et al., 2026). MeanFlow (Geng et al., 2026a) and AnyFlow (Gu et al., 2026) further learn average velocities or flow maps over finite intervals for flexible-step generation. Unlike their derivative- or finite-difference-based supervision, we construct finite-interval targets from frozen teacher trajectories and align training intervals with candidate inference schedules for tunable-budget WAM distillation.

**Trajectory geometry and adaptive computation.** Denoising trajectory geometry reflects both generative uncertainty and numerical difficulty. Concentrated posteriors yield more consistent velocities and straighter trajectories, whereas posterior dispersion and multimodality induce larger velocity variation and curvature (Yang et al., 2026; Rao et al., 2026); highly curved trajectories also incur larger integration errors under coarse discretization (Lee et al., 2023). These observations motivate adaptive computation according to denoising dynamics. Existing methods adjust denoising effort, network depth, or computation reuse using trajectory statistics, input-conditioned gating, or learned policies (Hu et al., 2024; Han et al., 2026; Zhang et al., 2025; Yu et al., 2025; Li et al., 2026a; Wang et al., 2026; Le et al., 2026). We use teacher-trajectory variation to characterize scene-dependent risk, while separately modeling the student's budget-dependent fidelity for computation allocation.

# 3 Preliminaries

## 3.1 World Action Models

Given an observation $\boldsymbol{o}$ and instruction $\boldsymbol{l}$, WAMs incorporate predictive visual modeling to generate an action chunk $\boldsymbol{a}_{1:H}$ of horizon $H$ (Bi et al., 2026; Li et al., 2026b; Yuan et al., 2026). We study their flow-based inference branches, which denoise actions and, where applicable, future visuals, without assuming a fixed video-action generation order. Appendix A details the different formulations.

## 3.2 Flow Matching and Flow Maps

**Flow matching.** Flow matching (Lipman et al., 2023) transports noise $\boldsymbol{x}_1$ to data $\boldsymbol{x}_0$ by solving $\mathrm{d}\boldsymbol{x}_t/\mathrm{d}t = \boldsymbol{v}(\boldsymbol{x}_t, t; \boldsymbol{c})$ from $t = 1$ to 0, where $\boldsymbol{v}$ is the velocity field conditioned on context $\boldsymbol{c}$.

**Flow maps.** A flow map instead represents a finite-time transition along the ODE trajectory. For $0 \le r < t \le 1$, its interval-average velocity is

$$\bar{\boldsymbol{v}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}) = \frac{1}{t-r}\int_r^t \boldsymbol{v}(\boldsymbol{x}_s, s; \boldsymbol{c})\,\mathrm{d}s. \tag{1}$$

A neural approximation $\boldsymbol{u}_{\boldsymbol{\theta}}$ parameterizes the transition as

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}) = \boldsymbol{x}_t - (t-r)\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}). \tag{2}$$

Conditioning on both endpoints supports different transition intervals and inference budgets.

**Derivative-based flow-map supervision.** MeanFlow (Geng et al., 2026a) constructs a target relating average and instantaneous velocities:

$$\boldsymbol{u}_{\mathrm{tgt}} = \boldsymbol{v}(\boldsymbol{x}_t, t; \boldsymbol{c}) - (t-r)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}), \tag{3}$$

and optimizes

$$\mathcal{L}_{\mathrm{MF}} = \mathbb{E}\left[\left\|\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}) - \mathrm{sg}(\boldsymbol{u}_{\mathrm{tgt}})\right\|_2^2\right], \tag{4}$$

where sg denotes stop-gradient. The total derivative follows the ODE trajectory with $r$ fixed. MeanFlow computes it through a Jacobian-vector product, whereas AnyFlow (Gu et al., 2026) uses central finite differences. These targets depend on the evolving student and can be sensitive to prediction and optimization noise (Geng et al., 2026b; Tu et al., 2026; Gu et al., 2026). Appendix B provides the full objectives and further discussion.

# 4 Method

Our framework (Figure 2) combines an AnyStep WAM for tunable-budget generation with a lightweight risk-benefit scheduler. We first adapt a pretrained WAM using integral-supervised flow

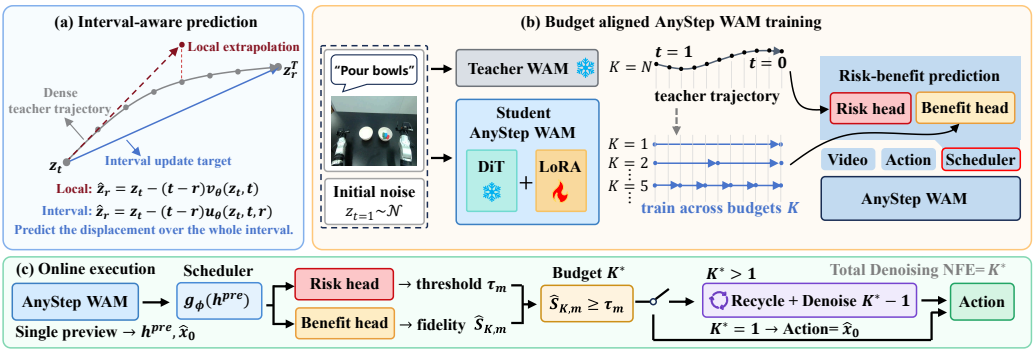


Figure 2: **Budget-aligned distillation and adaptive inference.** (a) Teacher-trajectory displacements supervise finite-interval updates. (b) Shared LoRA adapters support multiple budgets, while teacher dynamics and student-teacher agreement provide offline risk and benefit targets. (c) Single-preview scheduling and recycling require exactly $K^\star$ denoising evaluations, without online teacher access or candidate rollouts.

maps, then train the scheduler on offline teacher-student evaluations to predict chunk difficulty and budget-dependent fidelity. At inference, a single preview supplies features for budget selection and is reused in the selected trajectory. Here, $m \in \mathcal{M}$ indexes denoised modalities, with $\mathcal{M} = \{\mathrm{z}, \mathrm{a}\}$ for joint video-action models and $\mathcal{M} = \{\mathrm{a}\}$ for action-only inference.

### 4.1 Enabling AnyStep WAMs with Integral Flow-Map Distillation

We adapt the denoising branches of a pretrained WAM to the flow-map parameterization in Eq. 2, enabling a shared model to predict finite-time transitions under different denoising budgets. We train these transitions using explicit teacher-integrated targets rather than derivative-based supervision.

**Integral supervision from teacher trajectories.** Given a condition $\boldsymbol{c} = (\boldsymbol{o}, \boldsymbol{l})$ and initial noise $\boldsymbol{x}_1 = \{\boldsymbol{x}_1^m, m \in \mathcal{M}\}$, we first run the frozen teacher WAM using a fine-grained reference solver. Let $\boldsymbol{x}_t^{\mathrm{T},m}$ denote the teacher state of modality $m$ at denoising time $t$, with $\boldsymbol{x}_1^{\mathrm{T},m} = \boldsymbol{x}_1^m$. For two states on the same trajectory with $r < t$, we define the teacher average velocity of modality $m$ over the interval $[r, t]$ as

$$\overline{\boldsymbol{v}}_{t,r}^{\mathrm{T},m} = \frac{\boldsymbol{x}_t^{\mathrm{T},m} - \boldsymbol{x}_r^{\mathrm{T},m}}{t - r}, \qquad r < t. \tag{5}$$

This finite-displacement target directly approximates the integral average velocity in Eq. 1. Importantly, because both endpoints are determined by the frozen teacher, the target is independent of the student parameters and does not require derivative-based supervision through the student dynamics.

**Budget-aligned interval sampling.** We consider a set of candidate inference budgets $\mathcal{K}$. Each budget $K \in \mathcal{K}$ is associated with a fixed inference schedule $\mathcal{S}_K = \left\{1 = t_0^{(K)} > t_1^{(K)} > \cdots > t_K^{(K)} = 0\right\}$. Since different budgets induce different transition intervals, we align training with the transitions that the model will encounter at inference. Specifically, we first sample a budget $K \in \mathcal{K}$ and then an interval index $i \in \{0, \ldots, K-1\}$, yielding $(t, r) = (t_i^{(K)}, t_{i+1}^{(K)})$. The corresponding teacher state $\boldsymbol{x}_t^{\mathrm{T}}$ and average velocity $\overline{\boldsymbol{v}}_{t,r}^{\mathrm{T}}$ provide supervision for this transition, exposing the student to the finite-time intervals used by the candidate inference schedules.

**Training objective.** We train the student on these budget-aligned intervals and complement this supervision with local, endpoint, and recovery terms. All four terms share a dimension-normalized regression form. For an input state $\boldsymbol{x}_t = \{\boldsymbol{x}_t^m\}_{m\in\mathcal{M}}$ and target velocities $\boldsymbol{v}^\star = \{\boldsymbol{v}^{\star,m}\}_{m\in\mathcal{M}}$, we define

$$\mathcal{R}(\boldsymbol{x}_t, t, r, \boldsymbol{c}; \boldsymbol{v}^\star) = \sum_{m\in\mathcal{M}} \frac{1}{d_m} \left\| \boldsymbol{u}_{\boldsymbol{\theta}}^m(\boldsymbol{x}_t, t, r; \boldsymbol{c}) - \boldsymbol{v}^{\star,m} \right\|_2^2, \tag{6}$$

where $\boldsymbol{u}_{\boldsymbol{\theta}}^m$ denotes the modality-$m$ output conditioned on the joint input state, and $d_m$ is its output dimensionality. The normalization removes the direct dependence of each modality's loss contribution on its dimensionality. To align training with the preview-recycled inference procedure described in Section 4.3, the recovery loss uses a detached student-generated state $\bar{\boldsymbol{x}}_t = \mathrm{sg}(\widetilde{\boldsymbol{x}}_t^{(K)})$. For each

$K > 1$, we restart the frozen teacher once from the detached first recycled state and integrate to time zero, obtaining a budget-specific reference trajectory $\boldsymbol{x}_s^{\mathrm{T},K}$ for all remaining intervals. We define the corrective target $\boldsymbol{v}_{t,r}^{\mathrm{rec}} = (\bar{\boldsymbol{x}}_t - \boldsymbol{x}_r^{\mathrm{T},K})/(t-r)$ using this restarted reference, with no gradient through the target. Using this regression form, the complete training objective is

$$\mathcal{L}_{\mathrm{WAM}} = \lambda_{\mathrm{int}} \underbrace{\mathbb{E}_{\boldsymbol{c},\boldsymbol{x}_1,K,i}\left[\mathcal{R}(\boldsymbol{x}_t^{\mathrm{T}}, t, r, \boldsymbol{c}; \overline{\boldsymbol{v}}_{t,r}^{\mathrm{T}})\right]}_{\mathcal{L}_{\mathrm{int}}} + \lambda_{\mathrm{FM}} \underbrace{\mathbb{E}_{\boldsymbol{c},\boldsymbol{x}_0,\boldsymbol{x}_1,t}\left[\mathcal{R}(\boldsymbol{x}_t, t, t, \boldsymbol{c}; \boldsymbol{x}_1 - \boldsymbol{x}_0)\right]}_{\mathcal{L}_{\mathrm{FM}}}$$
$$+ \lambda_{\mathrm{end}} \underbrace{\mathbb{E}_{\boldsymbol{c},\boldsymbol{x}_1,t}\left[\mathcal{R}(\boldsymbol{x}_t^{\mathrm{T}}, t, 0, \boldsymbol{c}; \overline{\boldsymbol{v}}_{t,0}^{\mathrm{T}})\right]}_{\mathcal{L}_{\mathrm{end}}} + \lambda_{\mathrm{rec}} \underbrace{\mathbb{E}_{\boldsymbol{c},\boldsymbol{x}_1,K,i}\left[\mathcal{R}(\bar{\boldsymbol{x}}_t, t, r, \boldsymbol{c}; \boldsymbol{v}_{t,r}^{\mathrm{rec}})\right]}_{\mathcal{L}_{\mathrm{rec}}}, \tag{7}$$

where $(t, r) = (t_i^{(K)}, t_{i+1}^{(K)})$ in $\mathcal{L}_{\mathrm{int}}$. This $\mathcal{L}_{\mathrm{int}}$ term learns the finite-time transitions specified by the candidate schedules. The local term $\mathcal{L}_{\mathrm{FM}}$ preserves instantaneous-velocity prediction using standard flow-matching targets with identical time inputs, $r = t$. The endpoint term $\mathcal{L}_{\mathrm{end}}$ supervises transitions to $r = 0$, with dedicated $(t, r) = (1, 0)$ supervision for the one-step preview. Finally, $\mathcal{L}_{\mathrm{rec}}$ trains continuation from detached states along preview-recycled student trajectories. For each $K > 1$, it averages over the remaining intervals $(t, r) = (t_i^{(K)}, t_{i+1}^{(K)})$, $i = 1, \ldots, K-1$. The corrective target accounts for deviations from the restarted reference, so exact matching reaches its next endpoint $\boldsymbol{x}_r^{\mathrm{T},K}$. This additional supervision aligns the student with the preview-reuse procedure in Section 4.3.

We implement this adaptation using LoRA (Hu et al., 2021) in selected attention, feed-forward, and time-conditioning projections. All pretrained weights remain frozen, and the same adapters are shared across budgets. For models that denoise only actions at inference (Yuan et al., 2026), we retain only the action losses in our distillation objective.

### 4.2 Self-Aware Risk-Benefit Prediction

Different interaction contexts may require different amounts of denoising computation. To guide budget selection without evaluating multiple candidates online, we train a lightweight predictor to estimate teacher-trajectory difficulty and budget-specific student fidelity from offline supervision.

**Risk supervision.** We use velocity variation along the original teacher trajectory initialized at $\boldsymbol{x}_1$ as a budget-independent proxy for denoising difficulty. Let $\boldsymbol{v}_{j,m}^{\mathrm{T}}$ denote the modality-$m$ teacher velocity at the $j$th reference-solver step, with $N$ steps in total. We measure directional disagreement and relative velocity change:

$$\kappa_m^{\mathrm{dir}} = \frac{1}{N-1}\sum_{j=1}^{N-1}\left[1 - \cos\left(\boldsymbol{v}_{j+1,m}^{\mathrm{T}}, \boldsymbol{v}_{j,m}^{\mathrm{T}}\right)\right], \; \kappa_m^{\mathrm{rel}} = \frac{1}{N-1}\sum_{j=1}^{N-1}\frac{\mathrm{RMS}\left(\boldsymbol{v}_{j+1,m}^{\mathrm{T}} - \boldsymbol{v}_{j,m}^{\mathrm{T}}\right)}{\max\left\{\mathrm{RMS}\left(\boldsymbol{v}_{j,m}^{\mathrm{T}}\right), \epsilon\right\}}. \tag{8}$$

Here, cosine similarity and root-mean-square (RMS) are computed over valid generated coordinates, with numerical safeguards $\epsilon > 0$. For each modality, we average the training-set percentile ranks of the two statistics to obtain $\rho_m = \frac{1}{2}[F_m^{\mathrm{dir}}(\kappa_m^{\mathrm{dir}}) + F_m^{\mathrm{rel}}(\kappa_m^{\mathrm{rel}})]$, where $F_m^{\mathrm{dir}}$ and $F_m^{\mathrm{rel}}$ are empirical percentile mappings to $[0, 1]$. The resulting $\rho_m$ represents relative denoising difficulty rather than task-failure probability.

**Budget benefit supervision.** Risk characterizes teacher-trajectory variability but does not directly reflect the student's approximation error at a given budget. For each candidate budget $K \in \mathcal{K}$, we evaluate offline the same preview-recycled denoising path used at deployment. Let $\widetilde{\boldsymbol{x}}_{t_i^{(K)}}^{(K)}$ denote the student state along this path. The first interval uses the preview velocity $\boldsymbol{v}_{K,0}^{\mathrm{exec},m} = \boldsymbol{u}_{\boldsymbol{\theta}}^m(\boldsymbol{x}_1, 1, 0; \boldsymbol{c})$, while subsequent intervals use $\boldsymbol{v}_{K,i}^{\mathrm{exec},m} = \boldsymbol{u}_{\boldsymbol{\theta}}^m(\widetilde{\boldsymbol{x}}_{t_i^{(K)}}^{(K)}, t_i^{(K)}, t_{i+1}^{(K)}; \boldsymbol{c})$, $i = 1, \ldots, K-1$. Preview recycling scales the displacement by $1 - t_1^{(K)}$, so the average student velocity over the first interval remains the shared preview velocity. We compare it with $\boldsymbol{v}_{K,0}^{\mathrm{rec},m} = \overline{\boldsymbol{v}}_{1,t_1^{(K)}}^{\mathrm{T},m}$ over the same interval along the original teacher trajectory. For $i \geq 1$, $\boldsymbol{v}_{K,i}^{\mathrm{rec},m}$ is the modality-$m$ corrective target defined above, evaluated at $(t, r) = (t_i^{(K)}, t_{i+1}^{(K)})$ using the teacher trajectory restarted once from the recycled state. For $K = 1$, only the first interval applies, with teacher endpoint $\boldsymbol{x}_0^{\mathrm{T}}$.

$$S_{K,m} = \sum_{i=0}^{K-1} w_i^{(K)} \psi\left(\boldsymbol{v}_{K,i}^{\mathrm{exec},m}, \boldsymbol{v}_{K,i}^{\mathrm{rec},m}\right), \qquad \sum_{i=0}^{K-1} w_i^{(K)} = 1. \tag{9}$$

Here, $\psi$ combines directional and normalized velocity agreement:

$$\psi(\mathbf{p},\mathbf{q}) = \frac{1+\cos(\mathbf{p},\mathbf{q})}{4} + \frac{1}{2}\exp\left(-\frac{2\,\mathrm{RMS}(\mathbf{p}-\mathbf{q})}{\max\{\mathrm{RMS}(\mathbf{p})+\mathrm{RMS}(\mathbf{q}),\epsilon\}}\right).$$

The weights $w_i^{(K)} = 2^{-i}/\sum_{j=0}^{K-1} 2^{-j}$, $i = 0,\ldots,K-1$, emphasize earlier denoising intervals. Thus, $S_{K,m} \in [0,1]$ summarizes preview agreement and subsequent recovery fidelity. It remains a proxy rather than a direct measure of terminal action error or task success.

**Learning from one-step previews.** For the same condition and initial noise used to construct the supervision targets, a one-step evaluation $\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_1, 1, 0; \boldsymbol{c})$ provides hidden features and predicted velocities. Their summaries, together with contextual features, form the scheduler input $\mathbf{h}^{\mathrm{pre}}$. A lightweight predictor $g_\phi$ produces $g_\phi(\mathbf{h}^{\mathrm{pre}}) = \left(\{\widehat{\rho}_m\}_{m\in\mathcal{M}}, \{\widehat{S}_{K,m}\}_{K\in\mathcal{K},m\in\mathcal{M}}\right)$. Both prediction heads use sigmoid outputs and are supervised with Smooth L1 losses against the corresponding offline targets. Architectural details and training configurations are provided in Appendix C.

### 4.3 Risk-Conditioned Adaptive Inference

Adaptive inference uses predicted risk and budget-specific fidelity to select the smallest budget meeting risk-adaptive predicted requirements, then recycles the one-step preview to initialize the selected sampling schedule without candidate rollouts.

**Risk-conditioned budget selection.** At inference, a single evaluation produces the preview velocity $\boldsymbol{u}^{\mathrm{pre}} = \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_1, 1, 0; \boldsymbol{c})$, the prediction $\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_1 - \boldsymbol{u}^{\mathrm{pre}}$, and the features used by the risk-benefit predictor. For each active modality $m$, predicted risk interpolates between the easy and hard fidelity thresholds $\tau_m^{\mathrm{easy}}$ and $\tau_m^{\mathrm{hard}}$:

$$\tau_m = \tau_m^{\mathrm{easy}} + \left(\tau_m^{\mathrm{hard}} - \tau_m^{\mathrm{easy}}\right)\widehat{\rho}_m \tag{10}$$

Higher predicted difficulty therefore imposes a stricter fidelity requirement. We select the smallest budget $K^\star$ satisfying $\widehat{S}_{K^\star,m} \geq \tau_m$ for all active modalities, or the budget with the largest summed predicted benefit if none qualifies.

**Preview recycling.** As illustrated in Figure 2(c), the preview is reused within the selected budget. If $K^\star = 1$, we return $\widehat{\boldsymbol{x}}_0$ directly. Otherwise, we re-noise it to the first intermediate time using the original noise: $\widetilde{\boldsymbol{x}}_{t_1^{(K^\star)}}^{(K^\star)} = t_1^{(K^\star)}\boldsymbol{x}_1 + (1 - t_1^{(K^\star)})\widehat{\boldsymbol{x}}_0$. We then perform the remaining $K^\star - 1$ interval updates, requiring exactly $K^\star$ denoising evaluations including the preview, without online teacher access or candidate rollouts. The scheduler adds only lightweight prediction overhead.

## 5 Experiment

**Benchmarks.** We evaluate Motus (Bi et al., 2026), FastWAM (Yuan et al., 2026), and LingBotVA (Li et al., 2026b) through comparisons, ablations, and analyses. RoboTwin 2.0 (Chen et al., 2025) provides 50 bimanual manipulation tasks under clean and randomized settings, with randomization covering clutter, lighting, background textures, and tabletop heights. Real-world evaluation uses a Unitree G1D dual-arm robot with Unitree Dex1-1 grippers on six tasks: Bowl Pouring, Clean Table, Put Block, Put Cup, Stack Blocks, and Stack Bowls.

**Implementation and evaluation.** Each backbone is trained on four NVIDIA H100 GPUs for 8,000 steps with a global batch size of 32. RoboTwin evaluation uses a single NVIDIA A100 with 100 trials per task for every method and ablation variant; real-world inference runs locally on a single NVIDIA RTX 5090 with 20 trials per task on average. More details are in Appendix C. For budget selection, easy/hard thresholds use the 10th/20th percentiles of offline training-set video similarity and the 20th/80th percentiles of action similarity; Section 5.3 evaluates threshold sensitivity.

### 5.1 SIMULATION EXPERIMENTS

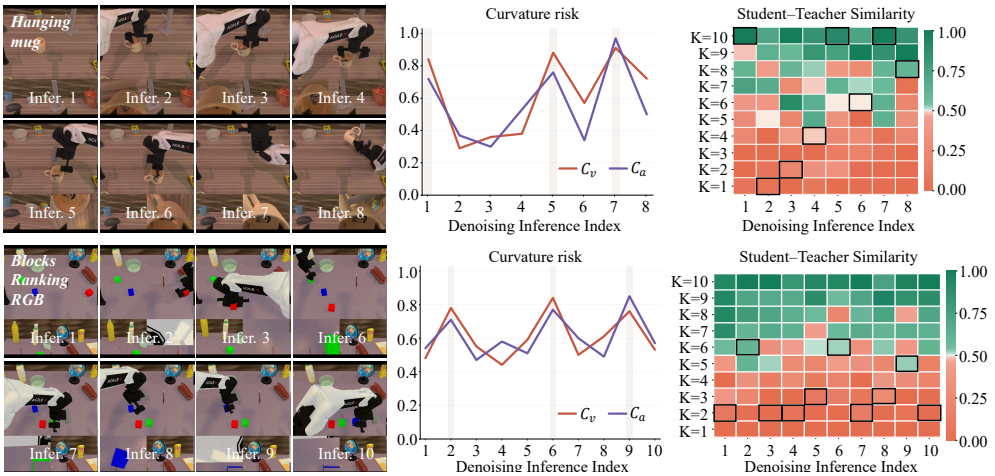


Figure 3: Adaptive denoising rollouts. The left column shows representative observations. The middle column plots predicted video ($C_v$) and action ($C_a$) curvature risks, shaded regions marking high-risk stages. The right column shows student-teacher velocity similarities, combined by their minimum. Black boxes mark selected budgets.

Table 1: RoboTwin 2.0 results. Base denotes the original model. SR denotes success rate (%); Steps denotes mean denoising steps for adaptive methods. Per-task results are in Appendix G.

| Method | Motus (Bi et al., 2026) | | | | | | FastWAM (Yuan et al., 2026) | | | | | | LingBotVA (Li et al., 2026b) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Clean | | Random | | Average | | Clean | | Random | | Average | | Clean | | Random | | Average | |
| | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps |
| Base | 76.00 | 1 | 74.10 | 1 | 75.05 | 1 | 77.26 | 1 | 77.14 | 1 | 77.20 | 1 | 74.36 | 1 | 73.18 | 1 | 73.77 | 1 |
| Flash-WAM | 79.20 | 1 | 78.66 | 1 | 78.93 | 1 | 82.26 | 1 | 81.14 | 1 | 81.70 | 1 | 82.56 | 1 | 80.26 | 1 | 81.41 | 1 |
| **AnyStep (Ours)** | **84.14** | 1 | **80.10** | 1 | **82.12** | 1 | **89.12** | 1 | **89.44** | 1 | **89.28** | 1 | **83.80** | 1 | **81.62** | 1 | **82.71** | 1 |
| Base | **88.66** | 10 | 87.02 | 10 | 87.84 | 10 | 91.88 | 10 | **91.78** | 10 | **91.83** | 10 | **92.93** | 25/50 | 91.55 | 25/50 | **92.24** | 25/50 |
| AnyStep+Gate | 87.60 | 5.23 | 86.92 | 4.95 | 87.26 | 5.09 | 90.28 | 6.34 | 90.04 | 6.98 | 90.16 | 6.66 | 90.16 | 4.58 | 90.52 | 5.04 | 90.34 | 4.81 |
| **AnyStep (Ours)** | 88.12 | **4.02** | **87.80** | **3.94** | **87.96** | **3.98** | **92.12** | **5.02** | 91.06 | **5.03** | 91.59 | **5.03** | 92.74 | **3.54** | **91.70** | **3.82** | 92.22 | **3.68** |

**Results on RoboTwin.** In Figure 3, contact-intensive stages tend to exhibit higher predicted risk, favoring larger denoising budgets that meet stricter fidelity requirements. Table 1 compares task success rates and generation efficiency across three WAM backbones. Base Motus and FastWAM use a fixed 10-step budget, whereas the serial LingBotVA architecture uses 25 steps for its video DiT and 50 for its action DiT. **Ours** maintains average SR within 0.24 percentage points of the baselines while selecting mean budgets of 3.98, 5.03, and 3.68 steps, respectively. These correspond to reductions of 60.2% for Motus, 49.8% for FastWAM, and 85.28%/92.64% for LingBotVA's video/action branches. On a single NVIDIA A100 GPU, average per-inference-call latency drops from 1.932 to 0.859 s for Motus, 0.496 to 0.295 s for FastWAM, and 9.732 to 3.247 s for LingBotVA, with respective budget-selection module overheads of only 2.881, 2.4996, and 2.9772 ms per prediction.

To the best of our knowledge, prior work has not specifically addressed adaptive budget selection for flow-map-trained WAMs. We therefore construct **AnyStep+Gate**, which applies training-free percentile gating to action-coordination and visual-temporal scores on the same distilled models (Appendix D). Our scheduler improves average SR over gating by 0.70, 1.43, and 1.88 percentage points while reducing average denoising steps by 21.8%, 24.5%, and 23.5% on Motus, FastWAM, and LingBotVA, respectively, supporting fidelity-aware allocation. Under a matched one-step denoising budget, **AnyStep 1 step** improves average SR over **Base 1 step** by 7.07, 12.08, and 8.94 percentage points on Motus, FastWAM, and LingBotVA, respectively. It also outperforms **Flash-WAM** (Akbari et al., 2026) by 3.19, 7.58, and 1.30 percentage points, respectively, supporting the effectiveness of our teacher-trajectory distillation for one-step action prediction across architectures.

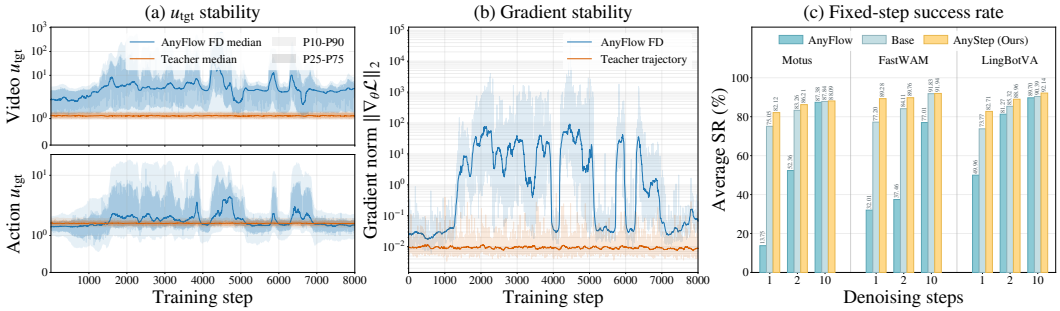


Figure 4: Training stability and fixed-budget performance. (a) Per-sample target RMS distributions. (b) Gradient norms. (c) Average success rates at fixed denoising steps.

**Fixed-budget generation and training stability.** Figure 4 compares finite-difference and teacher-trajectory supervision. Finite-difference targets exhibit wider sample-level distributions and produce frequent gradient spikes, with $45.9\%$ of steps exceeding the clipping threshold of $1.0$. Frozen-teacher targets remain concentrated and yield substantially more stable gradients (Fig. 4(a-b)). This stability also improves fixed-budget performance (Fig. 4(c)). At one denoising step, AnyStep improves average SR over Base by $7.07$, $12.08$, and $8.94$ points on Motus, FastWAM, and LingBotVA, respectively, and remains best at 2 and 10 steps. AnyStep therefore provides stable budget-aligned supervision without requiring teacher evaluation at inference. Results for additional denoising budgets are provided in Appendix E.

### 5.2 Real-world experiments

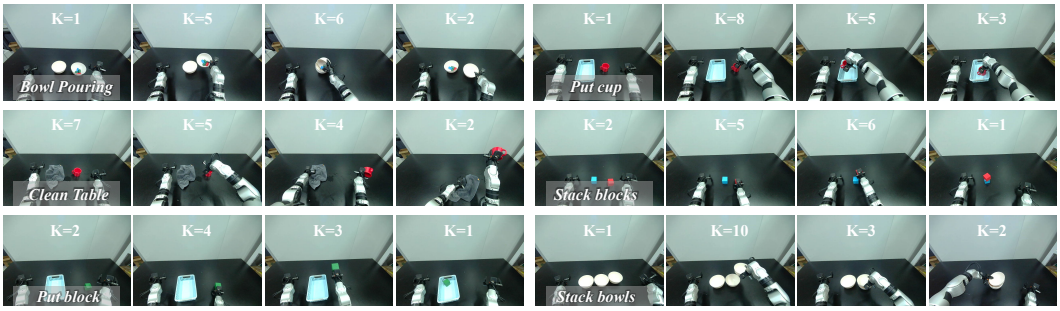


Figure 5: Real-world rollouts across six tasks on a Unitree G1D dual-arm robot. $K$ denotes the denoising budget adaptively selected for each shown inference call.

Table 2: Real-world results on six manipulation tasks.

| Task | Motus | | AnyStep-Motus | | FastWAM | | AnyStep-FastWAM | | LingBotVA | | AnyStep-LingBotVA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps |
| Bowl Pouring | 65 | 10 | 65 | 4.13 | 70 | 10 | 75 | 5.20 | 70 | 25/50 | 70 | 3.72 |
| Clean Table | 60 | 10 | 60 | 5.75 | 65 | 10 | 60 | 2.21 | 55 | 25/50 | 60 | 5.06 |
| Put Block | 70 | 10 | 75 | 2.56 | 75 | 10 | 70 | 3.54 | 75 | 25/50 | 70 | 2.21 |
| Put Cup | 70 | 10 | 75 | 3.95 | 75 | 10 | 85 | 2.31 | 70 | 25/50 | 75 | 3.47 |
| Stack Blocks | 55 | 10 | 60 | 2.81 | 50 | 10 | 55 | 4.62 | 60 | 25/50 | 60 | 4.45 |
| Stack Bowls | 75 | 10 | 80 | 5.17 | 85 | 10 | 85 | 3.77 | 80 | 25/50 | 85 | 2.31 |
| Avg. | 65.83 | 10 | **69.17** | **4.06** | 70.00 | 10 | **71.67** | **3.61** | 68.33 | 25/50 | **70.00** | **3.54** |

We evaluate six real-world manipulation tasks on a Unitree G1D dual-arm robot. Each base model and its AnyStep variant are trained using 50 episodes per task for 8,000 optimization steps on four NVIDIA H100 GPUs with a global batch size of 32. Figure 5 shows stage-dependent budgets, with more steps allocated to grasping and pouring in Bowl Pouring, illustrating how adaptive scheduling uses AnyStep's tunable-budget capability to refine demanding interactions. Table 2 shows average SR increasing to 69.17%, 71.67%, and 70.00% on Motus, FastWAM, and LingBotVA, while

mean denoising steps decrease by 59.4%, 63.9%, and 85.84%, respectively. On a single NVIDIA RTX 5090 GPU, average per-inference-call latency drops from 1.868 to 1.117 s for Motus, 0.280 to 0.122 s for FastWAM, and 4.654 to 0.758 s for LingBotVA, yielding respective speedups of $1.67\times$, $2.30\times$, and $6.14\times$. These results demonstrate faster inference with higher average task success.

### 5.3 Ablation experiments

Table 3 reports fixed-step and adaptive results for loss ablations, together with scheduler generalization and threshold sensitivity on the Motus backbone. Additional ablations on FastWAM and LingBotVA are reported in Appendix F. The **w/o $\mathcal{L}_{\mathbf{int}}$** variant removes budget-aligned teacher-interval supervision. Compared with **Full (Ours)**, its average one-step SR decreases by 1.40 percentage points, while two-step SR drops from 86.21% to 78.94% and adaptive SR from 87.96% to 78.09%, increasing the average budget from 3.98 to 5.27 steps. This highlights the importance of interval supervision for accurate tunable-budget prediction. The **w/o $\mathcal{L}_{\mathbf{end}}$** variant removes dedicated endpoint supervision, lowering one-step SR from 82.12% to 75.53% and increasing the adaptive budget to 4.74 steps. The **w/o $\mathcal{L}_{\mathbf{rec}}$** variant removes continuation supervision from recycled student states, reducing adaptive SR from 87.96% to 86.96% and supporting the contribution of recovery supervision to task success. **Clean-only** trains the scheduler without randomized samples. It achieves 87.56% SR with 4.05 steps under randomized conditions, close to **Full (Ours)** at 87.80% with 3.94 steps, supporting scheduler generalization beyond clean-only training conditions. Finally, **Lower thr.** and **Higher thr.** decrease and increase all default threshold percentile levels by five, respectively. They achieve average SRs of 87.60% and 87.93% with 3.08 and 5.85 steps. Both remain within 0.36 percentage points of **Full (Ours)**, indicating limited success-rate sensitivity over the tested threshold range while allowing substantial adjustment of computation.

Table 3: Ablations on RoboTwin 2.0. (a) Fixed-step SR (%); column numbers indicate denoising steps. (b) Adaptive SR (%) and mean denoising steps, including scheduler generalization and threshold sensitivity.

(a) Fixed-step evaluation

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | | w/o $\mathcal{L}_{\text{int}}$ | | | w/o $\mathcal{L}_{\text{rec}}$ | | | Full (Ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 |
| Clean | 88.66 | 76.70 | 86.94 | 87.40 | 82.18 | 79.66 | 80.20 | 84.20 | 86.30 | 87.96 | 84.14 | 87.50 | **88.78** |
| Random | 87.02 | 74.36 | 84.24 | 87.16 | 79.26 | 78.22 | 79.24 | 80.52 | 86.12 | **87.40** | 80.10 | 84.92 | **87.40** |
| Avg. | 87.84 | 75.53 | 85.59 | 87.28 | 80.72 | 78.94 | 79.72 | 82.36 | 86.21 | 87.68 | 82.12 | 86.21 | **88.09** |

(b) Adaptive inference and threshold sensitivity

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | w/o $\mathcal{L}_{\text{int}}$ | | w/o $\mathcal{L}_{\text{rec}}$ | | Clean-only | | Lower thr. | | Higher thr. | | Full (Ours) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps |
| Clean | **88.66** | 87.34 | 4.95 | 79.32 | 5.36 | 87.40 | 3.99 | 87.94 | 3.96 | 87.98 | 2.88 | 88.04 | 5.87 | 88.12 | 4.02 |
| Random | 87.02 | 87.10 | 4.52 | 76.86 | 5.17 | 86.52 | 3.45 | 87.56 | 4.05 | 87.22 | 3.27 | **87.82** | 5.83 | 87.80 | 3.94 |
| Avg. | 87.84 | 87.22 | 4.74 | 78.09 | 5.27 | 86.96 | 3.72 | 87.75 | 4.01 | 87.60 | 3.08 | 87.93 | 5.85 | **87.96** | 3.98 |

## 6 Conclusion

We presented AnyStep WAM for tunable-budget generation and adaptive computation allocation. Budget-aligned integral flow-map distillation learns finite-time transitions from frozen-teacher trajectories using shared LoRA adapters, without estimating student temporal derivatives. A lightweight scheduler selects the smallest predicted-feasible budget from difficulty and budget-dependent fidelity estimates, while preview recycling reuses the preview computation. Across Motus, FastWAM, and LingBotVA, AnyStep improves one-step success rates. Adaptive inference reduces denoising steps and latency on RoboTwin 2.0 and six real-world tasks while maintaining comparable or higher average success rates. These results support coupling tunable-budget generation with fidelity-aware scheduling for efficient WAM inference.

## AI USE STATEMENT

We used generative AI tools to assist with language polishing, including improving grammar, clarity, and readability. All AI-assisted edits were reviewed and revised by the authors. We take full responsibility for the final content of this paper, including its claims, results, and conclusions.

## REFERENCES


Arman Akbari, Ci Zhang, Arash Akbari, Lin Zhao, Yixiao Chen, Weiwei Chen, Xuan Zhang, Geng Yuan, and Yanzhi Wang. Flash-wam: Modality-aware distillation for world action models. *arXiv preprint arXiv:2606.05254*, 2026.

Sining Ang, Yuguang Yang, and Yan Wang. Adaptive-wam: Quality-guided early-exit planning from intermediate video-diffusion features. *arXiv preprint arXiv:2608.06008*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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.

Yitong Chen, Shiduo Zhang, Jingjing Gong, and Xipeng Qiu. Let it be simple: One-step action generation for vision-language-action models. *arXiv preprint arXiv:2606.05737*, 2026.

Yuxuan Gao, Shiqi Zhang, Yedong Shen, Yifan Duan, Wenhao Yu, Xin Zhang, Siyuan Cao, Jiajun Deng, and Yanyong Zhang. Driftingvla: Native one-step vision-language-action generation via per-dimension temporal drifting. *arXiv preprint arXiv:2608.29749*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models, 2026b. URL `https://arxiv.org/abs/2512.02012`.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation. In *European Conference on Computer Vision*, pp. 163–181. Springer, 2026.

Sunghwan Han, Youngtae Han, and Youngmin Yi. Adavla: Adaptive step flow matching for training-free acceleration of vision-language-action models. *arXiv preprint arXiv:2608.29208*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *Advances in Neural Information Processing Systems*, 37:138836–138858, 2024.

Thanh-Tung Le, Yunhan Zhao, Menglei Chai, Zhengyang Shen, Zhe Cao, Danhang Tang, Xiaohui Xie, and Deying Kong. Dsa: Dynamic step allocation for fast autoregressive video generation. *arXiv preprint arXiv:2606.04432*, 2026.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pp. 18957–18973. PMLR, 2023.

Andrew Zou Li, Gokul Swamy, Yonatan Bisk, and Andrea Bajcsy. Elastic: Efficiently learning to adaptively scale test-time compute for generative control policies. *arXiv preprint arXiv:2606.31132*, 2026a.

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*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Ziyang Rao, Yiren Zhao, Weiyu Guo, Ben Fei, Yandong Guo, and Hui Xiong. The geometry of flow-matching uncertainty: A cost-free uncertainty proxy and its application in flow-based vla failure detection, 2026. URL https://arxiv.org/abs/2607.27933.

Yuanpeng Tu, Yunpeng Chen, Xinyu Zhang, Chao Liao, and Hengshuang Zhao. Temporal equilibrium meanflow: Bridging the scale gap for one-step generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16064–16073, June 2026.

Ge Wang, Xinyu Tan, Xiang Li, Man Luo, Chengsi Yao, Shenhao Yan, Jiahao Yang, Fan Feng, Honghao Cai, Xiangyuan Wang, et al. Elastic queries reinforcement learning: Self-aware policy execution for vla models. *arXiv preprint arXiv:2606.14375*, 2026.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. Maniflow: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.

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.

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

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

Shu-Ang Yu, Feng Gao, Yi Wu, Chao Yu, and Yu Wang. D3p: Dynamic denoising diffusion policy via reinforcement learning. *arXiv preprint arXiv:2508.06804*, 2025.

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.

Hui Zhang, Zuxuan Wu, Zhen Xing, Jie Shao, and Yu-Gang Jiang. Adadiff: Adaptive step selection for fast diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9914–9922, 2025.

# A World Action Model Formulations

A visuomotor policy models $p(\boldsymbol{a}_{1:H} \mid \boldsymbol{o}, \boldsymbol{l})$. For WAMs that explicitly generate future visual states $\boldsymbol{z}_{1:T}$ and actions, this policy can be expressed as the marginal of their joint distribution:

$$p(\boldsymbol{a}_{1:H} \mid \boldsymbol{o}, \boldsymbol{l}) = \int p(\boldsymbol{a}_{1:H}, \boldsymbol{z}_{1:T} \mid \boldsymbol{o}, \boldsymbol{l}) \, \mathrm{d}\boldsymbol{z}_{1:T}. \tag{11}$$

This identity does not prescribe a generation order. Video and actions may be generated jointly (Bi et al., 2026) or through interleaved autoregressive prediction (Li et al., 2026b).

Alternatively, future visual prediction can serve as an auxiliary training objective, supporting action generation without explicit future-video denoising at inference (Yuan et al., 2026). Thus, predictive visual modeling during training does not necessarily imply video generation during deployment. Our framework accommodates these distinctions: adaptation and budget allocation apply to both video and action denoising when both are used at inference, and to action denoising alone otherwise.

# B Flow Matching and Flow Maps

**Flow matching.** Flow matching (FM) defines a continuous transport between noise and data distributions (Lipman et al., 2023). Let $\boldsymbol{x}_t$ denote the modeled state at $t \in [0, 1]$, where $\boldsymbol{x}_1$ is noise and $x_0$ is a data sample. Conditioned on context $\boldsymbol{c}$, the probability-flow ODE is

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}(\boldsymbol{x}_t, t; \boldsymbol{c}), \tag{12}$$

where $\boldsymbol{v}$ is the instantaneous velocity field. Standard generation solves Eq. 12 from $t = 1$ to $t = 0$ using multiple network evaluations.

**Flow maps and average velocity.** A flow map instead represents a finite-time transition along the same ODE trajectory. Let $\Phi_{r \leftarrow t}(\boldsymbol{x}_t) = \boldsymbol{x}_r$ for $1 \geq t \geq r \geq 0$. It satisfies $\Phi_{t \leftarrow t}(x) = x$ and $\Phi_{q \leftarrow r} \circ \Phi_{r \leftarrow t} = \Phi_{q \leftarrow t}$. The corresponding average velocity over $[r, t]$ is

$$\bar{\boldsymbol{v}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}) = \frac{1}{t - r} \int_r^t \boldsymbol{v}(\boldsymbol{x}_s, s; \boldsymbol{c}) \, ds, \tag{13}$$

such that $\Phi_{r \leftarrow t}(\boldsymbol{x}_t) = \boldsymbol{x}_t - (t - r)\bar{\boldsymbol{v}}(\boldsymbol{x}_t, t, r; \boldsymbol{c})$. A neural flow-map model therefore predicts an average velocity $u_\theta$ and parameterizes

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}) = \boldsymbol{x}_t - (t - r)\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}). \tag{14}$$

Conditioning on both endpoints $(t, r)$ allows one model to support different inference budgets.

**Derivative-based flow-map supervision.** MeanFlow (Geng et al., 2026a) derives a local identity relating average and instantaneous velocities,

$$\boldsymbol{u}_{\mathrm{tgt}} = \boldsymbol{v}(\boldsymbol{x}_t, t; \boldsymbol{c}) - (t - r)\frac{d}{dt}\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, r; \boldsymbol{c}), \tag{15}$$

and trains

$$\mathcal{L}_{\mathrm{MF}} = \mathbb{E}\left[\left\|\boldsymbol{u}_{\boldsymbol{\theta}} - \mathrm{sg}(\boldsymbol{u}_{\mathrm{tgt}})\right\|_2^2\right]. \tag{16}$$

The total derivative in Eq. 15 is computed through a Jacobian-vector product (JVP). AnyFlow (Gu et al., 2026) instead approximates it using a central finite difference to avoid explicit JVP computation. Both formulations construct the flow-map supervision through local derivative information of the learned average-velocity field. Such derivative-dependent objectives can be sensitive to optimization and prediction noise (Geng et al., 2026b; Tu et al., 2026; Gu et al., 2026).

# C Implementation Details

**RoboTwin simulation.** We use the official RoboTwin 2.0 demonstration dataset (Chen et al., 2025). For each of the 50 tasks, the dataset contains 50 demonstration episodes under the clean setting and 500 under the randomized setting. We instantiate AnyStep on three publicly available backbones: Motus (Bi et al., 2026), FastWAM (Yuan et al., 2026), and LingBotVA (Li et al., 2026b). For each backbone, we train a single multi-task AnyStep model shared across all 50 tasks. Clean and randomized demonstrations are sampled with equal probability during distillation.

**AnyStep adaptation.** During AnyStep distillation, all original backbone and vision-language model parameters remain frozen, and only the introduced LoRA adapters are optimized. The same adapters are shared across tasks and denoising budgets. We set the LoRA rank to 128 for all adapted layers across the three backbones. The training objective combines instantaneous-velocity, endpoint, budget-aligned interval, and recycled-state recovery supervision, with weights $\lambda_{\mathrm{FM}} = 0.20$, $\lambda_{\mathrm{end}} = 0.50$, $\lambda_{\mathrm{int}} = 0.30$, and $\lambda_{\mathrm{rec}} = 0.20$, respectively. We use these weights for all three backbones. Table 4 summarizes the distillation hyperparameters and trainable parameter counts.

Table 4: Hyperparameters and trainable parameter counts for LoRA-based AnyStep distillation. All original model parameters remain frozen during this stage.

| Configuration | Value |
|---|---|
| Optimizer | AdamW |
| Learning rate | $2 \times 10^{-5}$ |
| Weight decay | 0.01 |
| Learning-rate schedule | Linear schedule |
| Training iterations | 8,000 |
| Training GPUs | 4× NVIDIA H100 |
| Batch size per GPU | 8 |
| Global batch size | 32 |
| LoRA rank | 128 |
| $\lambda_{\mathrm{FM}}$ | 0.20 |
| $\lambda_{\mathrm{end}}$ | 0.50 |
| $\lambda_{\mathrm{int}}$ | 0.30 |
| $\lambda_{\mathrm{rec}}$ | 0.20 |
| Trainable LoRA parameters (Motus) | 286.39M |
| Trainable LoRA parameters (FastWAM) | 104.40M |
| Trainable LoRA parameters (LingBotVA) | 232.91M |

Table 5: Training hyperparameters of the adaptive scheduler. The teacher and AnyStep student remain frozen during scheduler training.

| Configuration | Value |
|---|---|
| Optimizer | AdamW |
| Learning rate | $1 \times 10^{-4}$ |
| Weight decay | 0.01 |
| Learning-rate schedule | Cosine decay |
| Number of training GPUs | 4 |
| Batch size per GPU | 64 |
| Global batch size | 256 |
| Training iterations | 4,000 |
| Prediction loss | Smooth L1 |
| Risk prediction loss weight | 1.0 |
| Fidelity prediction loss weight | 1.0 |

**Real-world experiments.** Our real-world dataset covers six manipulation tasks, with 50 demonstrations collected per task, yielding 300 demonstrations in total. For each backbone, we first train a shared multi-task base model using demonstrations from all six tasks. We then freeze the resulting model and optimize only the newly introduced LoRA adapters during AnyStep adaptation. The base model and its AnyStep variant use the same demonstration dataset. Video demonstrations of the real-world model deployment tests are provided in the supplementary material, all shown at 1× (real-time) speed.

**Candidate budgets and backbone-specific execution.** We use the candidate budget set $\mathcal{K} = \{1, 2, \ldots, 10\}$ for all three backbones. Motus denoises video and actions jointly, with both branches following a synchronized $K$-step schedule. FastWAM omits future-video denoising at inference, so adaptive budget selection and preview recycling apply only to its action branch. LingBotVA follows a sequential video-to-action procedure, which first denoises video and then generates the action conditioned on the video output.

**Adaptive scheduler data and training.** After AnyStep distillation, we freeze the student and construct the scheduler training set offline using the same training demonstrations. For each sampled context and initial noise, we extract features from the one-step preview and perform offline teacher-student evaluations to obtain teacher-derived risk labels and budget-dependent fidelity labels. Candidate budgets are evaluated using the preview-recycled execution procedure described in Section 4.3. This process requires no additional environment interaction or demonstration collection. Both the teacher and student remain frozen while the scheduler is trained to predict the offline labels using equally weighted Smooth L1 losses for risk and fidelity prediction. The scheduler architecture is described in Figure 6, and its training hyperparameters are listed in Table 5.

As shown in Figure 6, the scheduler predicts these offline targets from features available in a single one-step preview. For joint video-action models, its inputs comprise six feature groups: video/action final-block feature summaries, video/action predicted-velocity summaries, normalized robot state, a pooled instruction embedding. A four-layer Transformer processes the projected context tokens together with one risk query and $B = |\mathcal{K}|$ budget queries. The risk head predicts $\{\widehat{\rho}_m\}_{m\in\mathcal{M}}$, while the benefit head predicts a $B \times |\mathcal{M}|$ matrix $\{\widehat{S}_{K,m}\}_{K\in\mathcal{K},m\in\mathcal{M}}$. Both heads use sigmoid outputs and are trained with Smooth L1 losses against their respective offline targets. This allows a single preview to estimate difficulty and compare candidate budgets without online teacher evaluations or candidate rollouts.

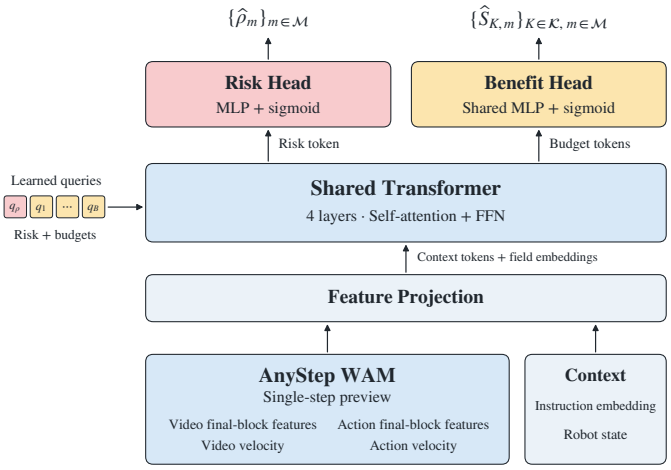


Figure 6: Internal architecture of the preview-based adaptive scheduler. Features and velocity predictions from a single-step AnyStep WAM preview are fused with instruction and robot-state context, projected into context tokens, and processed jointly with learned risk and budget queries by a shared Transformer. Query-specific heads predict modality-wise execution risk and the expected student-teacher similarity for each candidate denoising budget.

## D TRAINING-FREE GATING BASELINE

We first execute complete one-step action denoising as a probe. For each arm, we compute the relative contribution of each joint to the predicted motion,

$$p_{t,j} = \frac{|q_{t+1,j} - q_{t,j}|}{\sum_{k=1}^{6} |q_{t+1,k} - q_{t,k}| + \epsilon}, \tag{17}$$

and define the action-coordination metric as

$$C_{\text{raw}} = \max_{\text{arm}\in\{L,R\}} Q_{0.9}\left(\sum_{j=1}^{6} |p_{t+1,j} - p_{t,j}|\right). \tag{18}$$

Thus, $C_{\text{raw}}$ measures how strongly the dominant moving joints change within the one-step action proposal.

For the visual metric, we compute the patch-wise RGB difference $D(\mathbf{I}_a, \mathbf{I}_b)$ between two real observations as the 90-th percentile of the mean absolute differences over a $4 \times 4$ patch grid. Using the latest three observations, the visual irregularity is

$$V_{\text{raw}} = \frac{|D(\mathbf{I}_{t-1}, \mathbf{I}_t) - D(\mathbf{I}_{t-2}, \mathbf{I}_{t-1})|}{\frac{1}{2}\left[D(\mathbf{I}_{t-1}, \mathbf{I}_t) + D(\mathbf{I}_{t-2}, \mathbf{I}_{t-1})\right] + \epsilon}. \tag{19}$$

The two metrics are normalized using frozen ECDFs:

$$C = F_C(C_{\text{raw}}), \qquad V = 1 - F_V(V_{\text{raw}}), \qquad S_{CV} = \frac{1}{2}C + \frac{1}{2}V. \tag{20}$$

Finally, with $u = F_{CV}(S_{CV})$, the denoising budget is selected as

$$N = \min\left(K,\ \lfloor K u^{\gamma} \rfloor + 1\right), \tag{21}$$

where $K = 10$ and $\gamma = 1.6965$. If $N = 1$, the probe is directly reused; otherwise, a complete $N$-step trajectory is rerun from the same initial noise.

## E FIXED DENOISE STEP RESULTS ON ROBOTWIN

Table 6: Complete fixed-step SR (%) on RoboTwin 2.0. Base denotes the original models; AnyFlow and AnyStep denote the corresponding training methods. Avg. averages the clean and randomized settings.

| Steps | Method | Motus | | | FastWAM | | | LingBotVA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Clean | Random | Avg. | Clean | Random | Avg. | Clean | Random | Avg. |
| 1 | Base | 76.00 | 74.10 | 75.05 | 77.26 | 77.14 | 77.20 | 74.36 | 73.18 | 73.77 |
| | AnyFlow | 13.20 | 14.30 | 13.75 | 32.60 | 31.42 | 32.01 | 51.46 | 48.46 | 49.96 |
| | AnyStep (Ours) | **84.14** | **80.10** | **82.12** | **89.12** | **89.44** | **89.28** | **83.80** | **81.62** | **82.71** |
| 2 | Base | 82.60 | 83.92 | 83.26 | 84.52 | 83.70 | 84.11 | 85.38 | 85.26 | 85.32 |
| | AnyFlow | 52.92 | 51.80 | 52.36 | 38.24 | 36.68 | 37.46 | 83.28 | 79.26 | 81.27 |
| | AnyStep (Ours) | **87.50** | **84.92** | **86.21** | **90.20** | **89.32** | **89.76** | **88.64** | **89.28** | **88.96** |
| 4 | Base | 84.20 | 83.60 | 83.90 | 87.80 | 86.28 | 87.04 | 86.92 | 85.38 | 86.15 |
| | AnyFlow | 81.34 | 81.80 | 81.57 | 44.44 | 40.66 | 42.55 | 85.24 | 83.32 | 84.28 |
| | AnyStep (Ours) | **87.40** | **86.68** | **87.04** | **90.72** | **90.40** | **90.56** | **89.88** | **89.26** | **89.57** |
| 6 | Base | 86.60 | 85.20 | 85.90 | 88.14 | 87.34 | 87.74 | 88.48 | 88.12 | 88.30 |
| | AnyFlow | 84.78 | 84.92 | 84.85 | 61.68 | 60.92 | 61.30 | 90.02 | 86.24 | 88.13 |
| | AnyStep (Ours) | **87.56** | **86.32** | **86.94** | **91.60** | **90.52** | **91.06** | **90.60** | **90.14** | **90.37** |
| 8 | Base | 85.60 | 85.60 | 85.60 | 90.16 | 89.62 | 89.89 | 90.62 | 89.86 | 90.24 |
| | AnyFlow | 87.44 | 86.30 | 86.87 | 62.68 | 64.64 | 63.66 | 90.26 | 87.20 | 88.73 |
| | AnyStep (Ours) | **87.94** | **87.34** | **87.64** | **92.24** | **91.76** | **92.00** | **91.84** | **90.72** | **91.28** |
| 10 | Base | 88.66 | 87.02 | 87.84 | 91.88 | **91.78** | 91.83 | 90.42 | 90.36 | 90.39 |
| | AnyFlow | 87.62 | 87.14 | 87.38 | 78.42 | 75.60 | 77.01 | 90.12 | 89.28 | 89.70 |
| | AnyStep (Ours) | **88.78** | **87.40** | **88.09** | **92.48** | 91.40 | **91.94** | **92.92** | **91.36** | **92.14** |

Table 6 reports additional results on RoboTwin 2.0 across fixed denoising budgets. AnyFlow exhibits a pronounced performance drop at small budgets. Action generation requires precise predictions, as

small errors during grasping or contact can lead to task failure. The training instability observed in our experiments may compromise the accuracy of large denoising updates, making this issue particularly severe under few-step inference. Increasing the budget enables smaller updates and further refinement, which partially mitigate these errors, consistent with AnyFlow's improving success rates as the number of steps increases.

## F Additional Ablations on FastWAM and LingBotVA

We extend the Motus ablations in Table 3 to FastWAM and LingBotVA using the same variant definitions. Avg. denotes the arithmetic mean of clean and randomized results. Lower thr. and Higher thr. shift all applicable default percentile thresholds by $-5$ and $+5$ percentile points, respectively. For FastWAM, these changes apply only to the action thresholds.

Table 7: Ablations on FastWAM using RoboTwin 2.0. (a) Fixed-step SR (%); column numbers indicate denoising steps. (b) Adaptive SR (%) and mean denoising steps. Bold marks the highest SR in each row within each panel.

(a) Fixed-step evaluation

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | | w/o $\mathcal{L}_{\text{int}}$ | | | w/o $\mathcal{L}_{\text{rec}}$ | | | Full (Ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 |
| Clean | 91.88 | 79.22 | 89.92 | 91.64 | 85.48 | 82.26 | 82.98 | 89.40 | 91.00 | 92.22 | 89.12 | 90.20 | **92.48** |
| Random | **91.78** | 76.34 | 89.04 | 91.26 | 84.32 | 80.44 | 81.04 | 89.28 | 89.96 | 91.54 | 89.44 | 89.32 | 91.40 |
| Avg. | 91.83 | 77.78 | 89.48 | 91.45 | 84.90 | 81.35 | 82.01 | 89.34 | 90.48 | 91.88 | 89.28 | 89.76 | **91.94** |

(b) Adaptive inference and threshold sensitivity

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | w/o $\mathcal{L}_{\text{int}}$ | | w/o $\mathcal{L}_{\text{rec}}$ | | Clean-only | | Lower thr. | | Higher thr. | | Full (Ours) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps |
| Clean | 91.88 | 91.54 | 5.21 | 82.54 | 6.43 | 91.86 | 4.67 | 91.94 | 4.99 | 92.08 | 3.92 | **92.14** | 6.11 | 92.12 | 5.02 |
| Random | **91.78** | 90.48 | 5.92 | 80.72 | 6.12 | 90.58 | 4.92 | 90.98 | 5.27 | 90.84 | 3.97 | 91.58 | 5.98 | 91.06 | 5.03 |
| Avg. | 91.83 | 91.01 | 5.57 | 81.63 | 6.28 | 91.22 | 4.80 | 91.46 | 5.13 | 91.46 | 3.95 | **91.86** | 6.05 | 91.59 | 5.03 |

Table 8: Ablations on LingBotVA using RoboTwin 2.0. (a) Fixed-step SR (%); column numbers indicate denoising steps. (b) Adaptive SR (%) and mean denoising steps. Bold marks the highest SR in each row within each panel. Base uses 25/50 video/action steps; AnyStep budgets apply to each branch.

(a) Fixed-step evaluation

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | | w/o $\mathcal{L}_{\text{int}}$ | | | w/o $\mathcal{L}_{\text{rec}}$ | | | Full (Ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25/50 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 | 1 | 2 | 10 |
| Clean | **92.93** | 75.24 | 87.94 | 91.58 | 80.28 | 76.62 | 78.84 | 82.16 | 87.32 | 91.86 | 83.80 | 88.64 | 92.92 |
| Random | **91.55** | 74.02 | 88.20 | 91.24 | 77.16 | 73.14 | 76.28 | 80.04 | 88.84 | 90.82 | 81.62 | 89.28 | 91.36 |
| Avg. | **92.24** | 74.63 | 88.07 | 91.41 | 78.72 | 74.88 | 77.56 | 81.10 | 88.08 | 91.34 | 82.71 | 88.96 | 92.14 |

(b) Adaptive inference and threshold sensitivity

| Setting | Base | w/o $\mathcal{L}_{\text{end}}$ | | w/o $\mathcal{L}_{\text{int}}$ | | w/o $\mathcal{L}_{\text{rec}}$ | | Clean-only | | Lower thr. | | Higher thr. | | Full (Ours) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps | SR | Steps |
| Clean | 92.93 | 91.32 | 4.59 | 80.76 | 5.14 | 92.08 | 3.76 | 92.18 | 3.37 | 91.98 | 2.98 | **92.98** | 5.79 | 92.74 | 3.54 |
| Random | 91.55 | 91.06 | 4.81 | 78.08 | 5.99 | 91.24 | 4.02 | 91.26 | 3.96 | 90.56 | 3.11 | **91.82** | 6.23 | 91.70 | 3.82 |
| Avg. | 92.24 | 91.19 | 4.70 | 79.42 | 5.57 | 91.66 | 3.89 | 91.72 | 3.67 | 91.27 | 3.05 | **92.40** | 6.01 | 92.22 | 3.68 |

## G RoboTwin Detailed Results

Tables 9, 10, and 11 present task-wise comparisons between the base models and our AnyStep variants with adaptive denoising. We report success rates under clean and randomized settings, together with the average denoising steps used by AnyStep. Following their original inference configurations, the base models use their full fixed denoising schedules: 10 steps for Motus (Bi et al., 2026), 10 steps for FastWAM (Yuan et al., 2026), and 25 and 50 steps for the video and action DiTs of LingBotVA (Li et al., 2026b), respectively. Our AnyStep variants adaptively select the denoising budget for each prediction chunk.

Table 9: **Task-wise comparison of Motus and AnyStep-Motus across 50 tasks.** We report success rates (SR, %) under clean and randomized (Rand.) settings, and average denoising steps (Steps) for AnyStep-Motus. The final row reports the unweighted mean across tasks. Bold indicates the highest SR within each row and setting, including ties.

| Task | Motus (Bi et al., 2026) | | AnyStep-Motus | | | |
|---|---|---|---|---|---|---|
| | Clean | Rand. | Clean | | Rand. | |
| | SR ↑ | SR ↑ | SR ↑ | Steps ↓ | SR ↑ | Steps ↓ |
| Adjust Bottle | **89** | 93 | **89** | 5.52 | **95** | 4.53 |
| Beat Block Hammer | 95 | 88 | **97** | 3.49 | **93** | 4.03 |
| Blocks Ranking RGB | 99 | **97** | **100** | 2.82 | 95 | 3.11 |
| Blocks Ranking Size | **75** | 63 | 74 | 3.83 | **71** | 3.79 |
| Click Alarmclock | **100** | **100** | **100** | 2.55 | **100** | 2.49 |
| Click Bell | **100** | **100** | **100** | 2.23 | **100** | 1.78 |
| Dump Bin Bigbin | **95** | **91** | **95** | 3.86 | 90 | 4.30 |
| Grab Roller | **100** | **100** | **100** | 5.39 | **100** | 5.40 |
| Handover Block | **86** | **73** | 78 | 5.87 | 69 | 4.97 |
| Handover Mic | 78 | 63 | **89** | 3.56 | **75** | 4.70 |
| Hanging Mug | **38** | 38 | 34 | 6.46 | **55** | 6.44 |
| Lift Pot | 96 | **99** | **98** | 4.58 | **99** | 4.57 |
| Move Can Pot | 34 | 74 | **45** | 3.44 | **75** | 3.47 |
| Move Pillbottle Pad | 93 | **96** | **94** | 3.72 | 93 | 3.37 |
| Move Playingcard Away | **100** | 96 | **100** | 3.91 | **98** | 4.24 |
| Move Stapler Pad | 83 | **85** | **87** | 3.72 | 83 | 3.52 |
| Open Laptop | 95 | 91 | **97** | 3.96 | **95** | 3.72 |
| Open Microwave | 95 | **91** | **96** | 3.57 | 85 | 3.37 |
| Pick Diverse Bottles | **90** | **91** | **90** | 4.59 | 81 | 4.87 |
| Pick Dual Bottles | **96** | **90** | **96** | 4.13 | **90** | 4.40 |
| Place A2B Left | **88** | 79 | 85 | 3.97 | **95** | 3.58 |
| Place A2B Right | **91** | 87 | 82 | 3.59 | **95** | 3.50 |
| Place Bread Basket | 91 | 94 | **94** | 3.96 | **95** | 3.60 |
| Place Bread Skillet | 86 | 83 | **87** | 5.27 | **88** | 4.89 |
| Place Burger Fries | **98** | **98** | 95 | 2.71 | **98** | 2.80 |
| Place Can Basket | **81** | **76** | 78 | 3.56 | 69 | 3.84 |
| Place Cans Plasticbox | **98** | **94** | 91 | 3.69 | 88 | 4.36 |
| Place Container Plate | **98** | 99 | **98** | 2.86 | **100** | 2.70 |
| Place Dual Shoes | **93** | **87** | 92 | 4.85 | **87** | 5.57 |
| Place Empty Cup | **99** | 98 | **99** | 2.94 | **99** | 2.54 |
| Place Fan | 91 | **87** | **94** | 4.02 | 85 | 4.05 |
| Place Mouse Pad | **66** | 68 | 58 | 3.98 | **75** | 3.70 |
| Place Object Basket | 81 | **87** | **84** | 4.78 | 86 | 4.54 |
| Place Object Scale | **88** | 85 | 81 | 5.14 | **89** | 4.30 |
| Place Object Stand | **98** | **97** | **98** | 3.06 | 95 | 3.34 |
| Place Phone Stand | 87 | 86 | **91** | 4.27 | **95** | 3.63 |
| Place Shoe | 99 | 97 | **100** | 3.07 | **100** | 2.87 |
| Press Stapler | 93 | **98** | **96** | 3.70 | **98** | 2.72 |
| Put Bottles Dustbin | 81 | 79 | **83** | 5.41 | **82** | 5.18 |
| Put Object Cabinet | **88** | **71** | 60 | 6.31 | 55 | 5.67 |
| Rotate Qrcode | **89** | **73** | 83 | 5.37 | 66 | 5.18 |
| Scan Object | 67 | **66** | **73** | 5.87 | 64 | 5.75 |
| Shake Bottle | **100** | 97 | **100** | 3.85 | **99** | 3.60 |
| Shake Bottle Horizontally | **100** | **98** | **100** | 3.78 | 95 | 4.22 |
| Stack Blocks Three | 91 | 95 | **97** | 2.73 | **96** | 2.78 |
| Stack Blocks Two | **100** | **98** | **100** | 2.34 | 97 | 2.69 |
| Stack Bowls Three | **79** | **87** | 75 | 4.15 | **87** | 4.32 |
| Stack Bowls Two | 98 | **98** | **99** | 2.92 | 97 | 3.45 |
| Stamp Seal | 93 | 92 | **97** | 2.95 | **97** | 3.01 |
| Turn Switch | **84** | **78** | 77 | 4.72 | 76 | 3.47 |
| **Average** | **88.66** | 87.02 | 88.12 | 4.02 | **87.8** | 3.94 |

Table 10: **Task-wise comparison of FastWAM and AnyStep-FastWAM across 50 tasks.** We report success rates (SR, %) under clean and randomized (Rand.) settings, and average denoising steps (Steps) for AnyStep-FastWAM. The final row reports the aggregate results. Bold indicates the highest SR within each row and setting, including ties.

| Task | FastWAM (Yuan et al., 2026) | | AnyStep-FastWAM | | | |
|---|---|---|---|---|---|---|
| | Clean | Rand. | Clean | | Rand. | |
| | SR ↑ | SR ↑ | SR ↑ | Steps ↓ | SR ↑ | Steps ↓ |
| Adjust Bottle | **100** | **100** | **100** | 5.57 | **100** | 6.18 |
| Beat Block Hammer | 99 | 97 | **100** | 4.50 | **100** | 4.81 |
| Blocks Ranking RGB | **100** | **100** | **100** | 4.00 | 98 | 4.15 |
| Blocks Ranking Size | **94** | **98** | 93 | 4.55 | 92 | 4.81 |
| Click Alarmclock | **100** | **100** | **100** | 5.82 | **100** | 5.53 |
| Click Bell | **100** | **100** | **100** | 5.26 | **100** | 5.18 |
| Dump Bin Bigbin | 97 | **96** | **99** | 4.83 | 95 | 4.72 |
| Grab Roller | **100** | **100** | **100** | 4.31 | **100** | 4.42 |
| Handover Block | **95** | **81** | **95** | 6.13 | **81** | 6.25 |
| Handover Mic | 99 | **100** | **100** | 6.75 | **100** | 6.23 |
| Hanging Mug | 58 | **62** | **63** | 6.22 | 61 | 6.93 |
| Lift Pot | **100** | **100** | **100** | 4.17 | **100** | 4.60 |
| Move Can Pot | **90** | 88 | 86 | 5.43 | **91** | 5.17 |
| Move Pillbottle Pad | **100** | 99 | **100** | 4.69 | **100** | 4.52 |
| Move Playingcard Away | **100** | **100** | **100** | 5.61 | **100** | 5.25 |
| Move Stapler Pad | 77 | 64 | **80** | 5.23 | **66** | 4.75 |
| Open Laptop | 98 | **100** | **99** | 4.51 | 99 | 4.09 |
| Open Microwave | 62 | **45** | **79** | 4.39 | 36 | 5.20 |
| Pick Diverse Bottles | 80 | 85 | **87** | 4.96 | **87** | 5.17 |
| Pick Dual Bottles | **100** | 96 | 99 | 4.84 | **98** | 5.27 |
| Place A2B Left | **95** | 93 | 93 | 5.39 | **95** | 5.30 |
| Place A2B Right | 93 | **99** | **94** | 5.01 | 95 | 5.23 |
| Place Bread Basket | 91 | **93** | **93** | 5.00 | **93** | 5.43 |
| Place Bread Skillet | 90 | **93** | **93** | 6.64 | 91 | 6.64 |
| Place Burger Fries | **96** | **99** | 94 | 4.84 | 97 | 5.29 |
| Place Can Basket | **71** | **69** | 68 | 5.71 | 65 | 5.19 |
| Place Cans Plasticbox | **99** | 96 | 98 | 4.75 | **98** | 4.74 |
| Place Container Plate | 96 | **100** | **100** | 4.79 | **100** | 4.60 |
| Place Dual Shoes | **94** | 88 | 91 | 5.26 | **90** | 5.67 |
| Place Empty Cup | **100** | **100** | 98 | 3.87 | **100** | 3.94 |
| Place Fan | 96 | **96** | **99** | 4.87 | 92 | 5.13 |
| Place Mouse Pad | 83 | **89** | **90** | 4.90 | 88 | 4.96 |
| Place Object Basket | **89** | **88** | 82 | 5.40 | 82 | 5.10 |
| Place Object Scale | **90** | **97** | 88 | 5.25 | 96 | 5.36 |
| Place Object Stand | **90** | 94 | 86 | 4.06 | **95** | 3.99 |
| Place Phone Stand | 97 | **99** | **98** | 5.52 | **99** | 5.59 |
| Place Shoe | 96 | **99** | **97** | 4.69 | 98 | 4.45 |
| Press Stapler | 90 | **97** | **94** | 4.41 | 96 | 4.00 |
| Put Bottles Dustbin | **95** | **90** | 87 | 5.82 | **90** | 4.87 |
| Put Object Cabinet | **94** | **89** | 85 | 5.33 | 87 | 5.07 |
| Rotate Qrcode | **93** | **89** | 92 | 5.47 | 88 | 5.49 |
| Scan Object | 89 | **92** | **93** | 6.86 | 89 | 6.27 |
| Shake Bottle | **100** | **100** | **100** | 4.78 | **100** | 4.72 |
| Shake Bottle Horizontally | **100** | **100** | **100** | 4.63 | **100** | 4.57 |
| Stack Blocks Three | **95** | **97** | 90 | 4.16 | **97** | 4.03 |
| Stack Blocks Two | **100** | **100** | **100** | 3.44 | 98 | 3.77 |
| Stack Bowls Three | **80** | **81** | 75 | 5.22 | 77 | 5.04 |
| Stack Bowls Two | 92 | **98** | **95** | 4.74 | 93 | 4.50 |
| Stamp Seal | **90** | **94** | 89 | 4.75 | 92 | 4.44 |
| Turn Switch | 61 | 59 | **64** | 3.67 | **68** | 4.89 |
| **Average** | 91.88 | **91.78** | **92.12** | 5.02 | 91.06 | 5.03 |

Table 11: **Task-wise comparison of LingBotVA and AnyStep-LingBotVA across 50 tasks.** We report success rates (SR, %) under clean and randomized (Rand.) settings, and average denoising steps (Steps) for AnyStep-LingBotVA. The final row reports the aggregate results. Bold indicates the highest SR within each row and setting, including ties.

| Task | LingBotVA (Li et al., 2026b) | | AnyStep-LingBotVA | | | |
|---|---|---|---|---|---|---|
| | Clean | Rand. | Clean | | Rand. | |
| | SR ↑ | SR ↑ | SR ↑ | Steps ↓ | SR ↑ | Steps ↓ |
| Adjust Bottle | 90 | 94 | **100** | 3.36 | **96** | 4.00 |
| Beat Block Hammer | 96 | **98** | **97** | 2.93 | **98** | 4.15 |
| Blocks Ranking RGB | **99** | **98** | 94 | 3.28 | 93 | 3.86 |
| Blocks Ranking Size | 94 | **96** | **97** | 3.27 | 81 | 3.81 |
| Click Alarmclock | 99 | **100** | **100** | 3.72 | **100** | 4.65 |
| Click Bell | **100** | **100** | **100** | 3.95 | **100** | 4.60 |
| Dump Bin Bigbin | 89 | 96 | **97** | 3.71 | **98** | 3.69 |
| Grab Roller | **100** | **100** | **100** | 3.34 | **100** | 4.22 |
| Handover Block | 99 | 78 | **100** | 3.46 | **95** | 3.43 |
| Handover Mic | 94 | **96** | **97** | 3.57 | 95 | 3.57 |
| Hanging Mug | **40** | 28 | 29 | 3.16 | **33** | 3.79 |
| Lift Pot | **100** | 99 | **100** | 2.80 | **100** | 3.99 |
| Move Can Pot | 94 | **97** | **97** | 3.27 | **97** | 3.89 |
| Move Pillbottle Pad | 99 | **99** | **100** | 3.25 | 98 | 3.88 |
| Move Playingcard Away | **100** | 99 | **100** | 3.61 | **100** | 4.25 |
| Move Stapler Pad | **91** | **79** | 67 | 3.57 | 71 | 3.56 |
| Open Laptop | 92 | **94** | **97** | 3.63 | **94** | 3.67 |
| Open Microwave | **82** | **86** | 69 | 3.14 | 85 | 3.82 |
| Pick Diverse Bottles | 89 | 82 | **92** | 3.87 | **89** | 3.66 |
| Pick Dual Bottles | **100** | **99** | **100** | 3.41 | 95 | 3.62 |
| Place A2B Left | 97 | **93** | **100** | 3.80 | 90 | 3.84 |
| Place A2B Right | **97** | **95** | **97** | 3.75 | 94 | 3.90 |
| Place Bread Basket | **97** | **95** | 94 | 3.66 | 90 | 3.54 |
| Place Bread Skillet | 95 | 90 | **97** | 3.87 | **95** | 3.75 |
| Place Burger Fries | **97** | 95 | **97** | 3.67 | **100** | 3.59 |
| Place Can Basket | 81 | 84 | **89** | 3.47 | **85** | 3.96 |
| Place Cans Plasticbox | **100** | **99** | **100** | 3.59 | 98 | 3.57 |
| Place Container Plate | **99** | **97** | 97 | 3.88 | 94 | 3.85 |
| Place Dual Shoes | **94** | **89** | 89 | 3.53 | 83 | 3.45 |
| Place Empty Cup | **100** | **100** | **100** | 3.68 | **100** | 3.85 |
| Place Fan | **99** | **93** | 97 | 3.86 | **93** | 3.70 |
| Place Mouse Pad | **93** | **96** | 92 | 3.73 | 94 | 3.84 |
| Place Object Basket | **91** | 88 | 88 | 3.45 | **93** | 3.45 |
| Place Object Scale | 96 | **96** | **97** | 3.27 | 92 | 3.85 |
| Place Object Stand | 99 | **96** | **100** | 3.88 | 94 | 3.88 |
| Place Phone Stand | 97 | **97** | **98** | 3.94 | 94 | 3.92 |
| Place Shoe | **98** | **98** | 96 | 3.87 | **98** | 3.88 |
| Press Stapler | **85** | 82 | 82 | 3.67 | **96** | 3.97 |
| Put Bottles Dustbin | **87** | **91** | 73 | 3.15 | 79 | 3.74 |
| Put Object Cabinet | 85 | 87 | **92** | 3.42 | **91** | 3.46 |
| Rotate Qrcode | 96 | 91 | **98** | 3.81 | **93** | 3.73 |
| Scan Object | **96** | **91** | 90 | 3.60 | **91** | 3.71 |
| Shake Bottle | **100** | 97 | **100** | 3.67 | **100** | 4.26 |
| Shake Bottle Horizontally | **100** | **99** | **100** | 3.60 | 98 | 4.11 |
| Stack Blocks Three | 99 | 98 | **100** | 3.31 | **100** | 3.91 |
| Stack Blocks Two | **100** | 98 | **100** | 3.48 | **100** | 3.49 |
| Stack Bowls Three | **86** | **83** | 84 | 3.26 | 79 | 3.79 |
| Stack Bowls Two | 94 | **98** | **100** | 3.48 | 96 | 3.44 |
| Stamp Seal | **96** | **97** | **96** | 3.85 | 96 | 3.93 |
| Turn Switch | 44 | 45 | **61** | 3.50 | **61** | 3.53 |
| **Average** | **92.93** | 91.55 | 92.74 | 3.54 | **91.7** | 3.82 |