StreamingWAM: Streaming World Action Modeling via Chunk-Wise Staircase Flow Matching

Anonymous authors
Paper under double-blind review

Abstract

World Action Models (WAMs), often built upon pretrained video-generation world models, have emerged as a promising framework for general-purpose and open-ended robot control. However, their real-world deployment often suffers from discontinuous and jerky execution. Existing methods typically address this issue by either improving inference efficiency or enhancing action continuity across prediction chunks, without jointly addressing both challenges. To bridge this gap, we propose StreamingWAM, a streaming-diffusion framework for WAM-based robot control. StreamingWAM combines a noise ladder for chunk-wise action prediction, a prefix rollout module for cross-chunk continuity, and dense video-history memory for temporally consistent conditioning. Experiments on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world robotic tasks demonstrate that StreamingWAM improves action continuity across static and dynamic settings, achieving competitive performance with the lowest response latency among the evaluated WAMs using the same backbone. Further analysis shows that smoother action transitions and faster responses to state changes contribute to higher success rates on the manipulation tasks, while the streaming staircase reduces sensitivity to the execution horizon K.

1 Introduction

Recent WAMs are built upon world models, particularly pretrained video-generation models (Ye et al., 2026b; Li et al., 2026b; Yuan et al., 2026; Kim et al., 2026). Through pretraining objectives based on future-state prediction, these models acquire knowledge of environment dynamics, which is then adapted to robotic domains via fine-tuning for general-purpose and open-ended robot control (Zhang et al., 2026; Wang et al., 2026a). By leveraging this learned dynamics prior, WAMs can exhibit strong generalization when deployed on real-world robots across diverse tasks and environments (Wang et al., 2026b).

Despite their impressive performance, WAMs often exhibit temporally discontinuous and jerky motions during closed-loop execution (Team, 2026; Black et al., 2025). This behavior is closely tied to their non-streaming inference procedure: at each replan, existing WAMs regenerate a complete action chunk from scratch through multi-step denoising and discard the unexecuted suffix of the previous plan. This procedure creates two coupled limitations. First, the combination of a large video-generation backbone, joint video–action processing, and iterative denoising makes WAM inference computationally expensive. Regenerating a complete action chunk at every replan further compounds this cost, resulting in substantial response latency and slower adaptation to new observations (Lu et al., 2026; Ma et al., 2026; Cai et al., 2026; Li et al., 2026a). Second, independently generated chunks may be inconsistent with previously executed actions, causing abrupt transitions at replanning boundaries (Lu et al., 2026; Høeg et al., 2024). Existing methods typically improve either inference efficiency or cross-chunk action continuity, but rarely both within a unified framework.

To bridge this gap, we propose StreamingWAM (Figure 1), which introduces a streaming diffusion paradigm into WAMs. Specifically, the **noise ladder** staggers future action blocks across noise levels in a persistent buffer, enabling low-latency, single-step steady-state replanning. The **prefix rollout module** conditions future-action denoising on a clean prefix of recently executed actions, improving continuity across action chunks. Finally, **dense video-history memory** provides recent visual context for action prediction, supporting temporally consistent responses to state changes.

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.

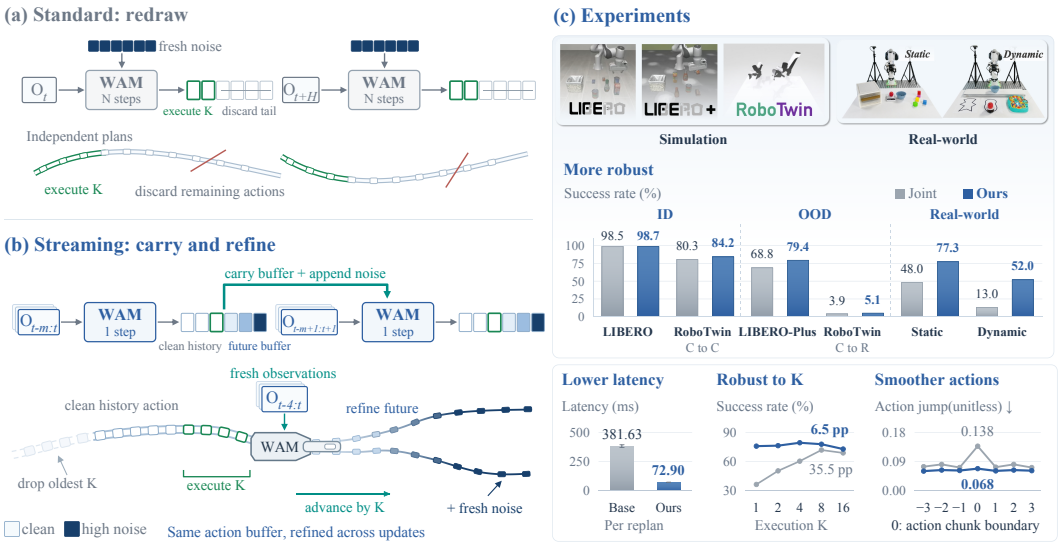


Figure 1: Overview of our streaming world-action model. (a) Standard policies regenerate a complete action chunk at every replan. (b) StreamingWAM rolls the action buffer forward with one denoising step per replan. (c) StreamingWAM performs strongly across simulation and real-world benchmarks.

We evaluate StreamingWAM on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world manipulation tasks spanning static and dynamic settings, where it achieves competitive success with the lowest end-to-end reaction latency among methods using the same backbone. Further analysis shows that the proposed components jointly improve cross-chunk action continuity, reduce sensitivity to the execution horizon K, and enable faster adaptation to changing observations.

Our contributions are summarized as follows:

- We introduce StreamingWAM, which integrates a noise ladder for chunk-wise streaming action prediction, a prefix rollout module for maintaining continuity with previously executed actions, and dense video-history memory for incorporating historical visual context into action prediction. Together, these components improve inference efficiency and execution continuity.
- We evaluate StreamingWAM on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world manipulation tasks spanning both static and dynamic settings. StreamingWAM achieves competitive task success and the lowest end-to-end reaction latency among the compared methods built on the same backbone. Extensive ablations further validate the contribution of each proposed component.
- Through controlled ablations and static and dynamic evaluations, we analyze the complementary effects of the proposed components. The noise ladder and prefix rollout improve cross-call action continuity, reduce sensitivity to the execution horizon K, and enable low-latency replanning. The clean action prefix and dense video history provide recent action–observation context, supporting temporally consistent execution and adaptation to object motion.

2 Related Work

World action models and vision-language-action models Two dominant paradigms have emerged for generalist robot control. Vision-language-action models (VLAs) such as π_0 (Black et al., 2024) and $\pi_{0.5}$ (Physical Intelligence et al., 2025) build on pretrained vision-language models, combining Internet-scale semantic knowledge with large-scale cross-embodiment robot training. WAMs instead build on pretrained video-generation models and jointly predict future visual states and actions, transferring video-derived dynamics priors to robot control. DreamZero (Ye et al., 2026b) and LingBot-VA (Li et al., 2026b) demonstrated the viability of this paradigm, achieving strong general-

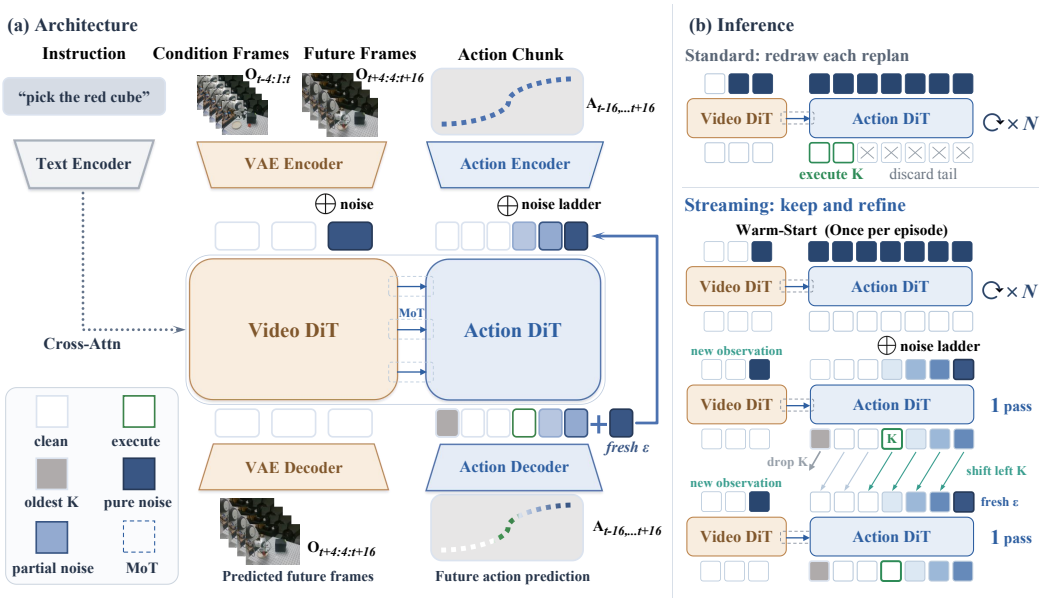


Figure 2: Overview of StreamingWAM. (a) Joint video–action architecture with a clean action prefix and staircase denoising. (b) Standard multi-step inference versus single-step streaming inference.

ization across environments and embodiments. However, large video backbones and iterative joint video–action denoising introduce substantial inference latency, limiting high-frequency closed-loop control.

Efficient world action models Recent work accelerates WAM inference through lightweight action modules, reduced video-side computation, asynchronous video–action denoising, and action-centered architectures (Yuan et al., 2026; Ma et al., 2026; Li et al., 2026a; Guo et al., 2026; Ye et al., 2026a). These methods mainly reduce per-step computation while retaining the standard full-chunk, multi-step replanning loop; in contrast, StreamingWAM carries and refines a persistent action buffer with a single forward pass per replan.

Diffusion policies and streaming diffusion Diffusion Policy generates action chunks through iterative denoising (Chi et al., 2023), whereas Streaming Diffusion Policy (SDP) and Diffusion Forcing use position- or token-dependent noise levels to support rolling and flexible-horizon generation (Høeg et al., 2024; Chen et al., 2024). StreamingWAM adapts this principle to WAMs, preserving the video-conditioned dynamics prior while enabling persistent, single-pass replanning.

3 Method

3.1 Problem formulation

At each planning step t, the robot queries the policy with the current visual observation o_t and task instruction l to obtain an H-step chunk of future actions $a_{t:t+H-1}$, where H denotes the action prediction horizon. A WAM realizes this policy by jointly denoising the action chunk and a short future video sequence $v_{t:t+T}$, where T is the video prediction horizon. At deployment, an execution horizon $K \leq H$ determines how many predicted actions are applied before replanning: the robot executes $a_{t:t+K-1}$, discards the unused suffix $a_{t+K:t+H-1}$, and replans when the next observation becomes available.

Streaming policy formulation. To enable streaming action control, we augment this standard WAM interface with temporal context and a persistent plan buffer. At each replanning time t, the model consumes the m most recent visual observations, the last P actions executed by the robot, and the unfinished plan carried over from the previous call. It returns the next K executable actions

together with an updated plan buffer for the subsequent call:

$$(a_{t:t+K-1}, \mathcal{B}_{t+K}) = \pi_\theta \left(\underbrace{o_{t-m+1:t}}_{\text{recent visual observations}}, l, \underbrace{a_{t-P:t-1}}_{\text{executed-action history}}, \underbrace{\mathcal{B}_t}_{\text{unfinished plan}} \right). \tag{1}$$

Here, t indexes environment steps, P is the executed-action history length, and $\mathcal{B}_t$ denotes the persistent plan buffer aligned with the current control step. The robot executes only $a_{t:t+K-1}$. Although the model maintains an H-step action plan internally, action slots beyond the executable prefix remain in $\mathcal{B}_{t+K}$ and are carried forward to the next planning call. These future actions may still be partially denoised and are never issued to the robot until they enter the executable prefix.

Standard WAM inference lacks cross-call plan persistence. At each replan, a standard WAM samples a new H-step action chunk from noise and discards the unexecuted suffix after executing the first K actions. Because all action slots follow the same denoising schedule, it neither reuses the previous plan nor maintains partially refined future actions across calls. The streaming interface in Equation (1) therefore requires a persistent plan buffer and a position-dependent denoising schedule.

3.2 StreamingWAM

Overview We propose StreamingWAM (Figure 2), a streaming diffusion formulation built on a Mixture-of-Transformers (MoT) WAM with dedicated video and action experts. It combines a **prefix rollout module** that conditions prediction on clean executed actions, a **noise ladder** that progressively refines a persistent future-action buffer, and **dense video-history memory** that supplies recent visual context. Together, these components amortize denoising across planning calls, reducing steady-state latency while improving cross-chunk continuity. StreamingWAM requires neither additional trainable parameters nor additional training data.

Prefix rollout module and dense video-history memory At each planning step t, we form an action sequence consisting of a clean prefix of P previously executed actions, followed by H future action slots The prefix is not a prediction target: during training, it is populated with ground-truth actions preceding the target chunk, whereas at deployment, it contains the actions actually executed by the robot. These prefix actions are provided without noise, allowing the action branch to model the future action sequence as

$$p_\theta(a_{t:t+H-1} \mid z(o_{t-m+1:t}, l),\, a_{t-P:t-1}). \tag{2}$$

Here, $o_{t-m+1:t}$ forms the dense video-history memory, capturing short-term visual dynamics that complement the executed-action prefix to improve the temporal consistency of action generation. As planning advances, newly executed actions are rolled into the conditioning prefix, while the unexecuted portion of B_t is shifted forward and reused. Thus, overlapping future actions are not regenerated from scratch, breaking the cross-call independence of standard WAM inference described in Section 3.1. Following Black et al. (2025), this prefix also provides longer-term context than the short visual-history window.

Streaming denoising over the future segment The clean prefix alone does not carry the future plan across calls, since the H future action slots would still be reinitialized from pure noise. We therefore assign them a staircase of noise levels. The future slots are grouped into consecutive blocks of width K, corresponding to the execution horizon, with all positions in each block sharing one noise level and later blocks assigned higher levels. Accordingly, after every K actions are executed, the prefix and the carried plan are rolled forward by exactly one block. Writing $c(j) = \lfloor (j-1)/K \rfloor$ for the block index of the j-th future position and $C = \lceil F/K \rceil$ for the number of blocks, the noise level at that position is

$$t_{P+j} = \frac{1}{N} \left(\left\lfloor \frac{N\,(c(j)+1)}{C} \right\rfloor - \delta \right), \qquad \delta \sim \mathcal{U}\{0, \ldots, \lfloor N/C \rfloor - 1\}, \tag{3}$$

where N is the number of noise levels and the random offset δ shifts the whole staircase down, exposing the model to the partially denoised states that occur midway through rolling execution.

Multi-width staircase training. The staircase is entirely determined by the block width K, and we do not hold it fixed. For each training example, we independently sample the execution horizon K uniformly from a finite candidate set $\mathcal{K}$ and construct the corresponding noise profile. Furthermore, We retain the standard shared-timestep objective for a fraction of training steps to preserve the model's multi-step denoising capability, which is required during the warm-up phase at the beginning of each episode, when the initial plan is generated from pure noise.

Training objective Both the video and action branches are trained with weighted flow matching. Given a clean target y, Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and a noise level $\tau \in [0, 1]$, we construct the interpolant $y_\tau = (1-\tau)y + \tau\epsilon$ and regress its velocity $\epsilon - y$:

$$\mathcal{L}_{vid} = \mathbb{E}_{y,\epsilon,\tau}\left[w(\tau)\,\|f_\theta(y_\tau, \tau, o, l) - (\epsilon - y)\|_2^2\right], \tag{4}$$

where $w(\tau)$ is inherited from the base model and emphasizes intermediate noise levels. For the action branch, our noise ladder assigns the H future actions a noise profile $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_H)$ defined by Equation (3). The action prefix a^{pre} is provided as clean conditioning and excluded from the loss. The action objective is

$$\mathcal{L}_{\text{act}} = \mathbb{E}\left[\frac{1}{H}\sum_{j=1}^{H} w(\tau_j)\left\|u_j^{\text{fut}} - \left(\epsilon_j - a_j^{\text{fut}}\right)\right\|_2^2\right], \qquad u^{\text{fut}} = f_\theta\left(a^{\text{pre}}, a_{\boldsymbol{\tau}}^{\text{fut}}, \tau, o, l\right). \tag{5}$$

Here, a_j^{fut} and u_j^{fut} denote the j-th future action chunk and its predicted velocity, respectively. Since τ_j varies across noise-ladder blocks, $w(\tau_j)$ produces a corresponding block-wise loss profile. The video objective remains unchanged, and the total loss is $\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda\mathcal{L}_{\text{vid}}$.

3.3 Inference Process

At deployment, the future plan is maintained in a persistent buffer $\mathcal{B}_t$ of H action slots. At each planning call, a single joint forward pass performs one Euler update toward zero noise. Let $\Delta > 0$ denote the Euler step size along the noise ladder. The buffer is updated and rolled as

$$\begin{gathered}\widetilde{\mathcal{B}}_t = \mathcal{B}_t - \Delta\, f_\theta^{\text{act}}(\mathcal{B}_t, \boldsymbol{\tau}, o_{t-m+1:t}, l, a_{t-P:t-1}), \qquad a_{t:t+K-1} = \widetilde{\mathcal{B}}_{t;1:K}, \\ \mathcal{B}_{t+K} = \text{Concat}\left(\widetilde{\mathcal{B}}_{t;K+1:H}, \epsilon^{\text{new}}\right).\end{gathered} \tag{6}$$

Here, $\widetilde{\mathcal{B}}_t$ is the buffer after the Euler update, and $\epsilon^{\text{new}} \sim \mathcal{N}(0, I)$ is a fresh block of K noisy actions. The leading block reaches zero noise and is executed, while the remaining $H-K$ actions are retained and shifted forward. The executed block is added to the clean action prefix for the next planning call but is not stored in $\mathcal{B}_{t+K}$.

Warm-up initialization. At the beginning of each episode, we perform a one-time warm-up to construct the initial persistent buffer $\mathcal{B}_0$. Specifically, we first generate a clean action chunk using the standard multi-step flow-matching sampler and then re-noise its blocks according to the staircase, following the denoise initialization strategy of Høeg et al. (2024). Since no executed-action history is available at this point, the action prefix a^{pre} is initialized with zeros. After this warm-up, each planning call follows the prefix-rollout procedure and requires only a single joint forward pass.

4 Experiments

4.1 Setup

Metrics. For most simulated and real-world tasks, we report the task success rate (%). For Hack-a-Mole, which requires repeated reactions rather than a single terminal success, we report the number of successful hits within a fixed time budget. We additionally report time to first action (TTFA) to characterize system responsiveness Lu et al. (2026). To assess action continuity, we report the per-step absolute position change, the displacement of the seven-dimensional joint configuration from its initial posture, and the corresponding acceleration.

Table 1: Success rates on LIBERO and LIBERO-PLUS. On LIBERO, we evaluate each task over 50 episodes (500 episodes per suite) and report the average success rate (%) across the four suites. On LIBERO-PLUS, we run one episode for each perturbed task instance, totaling 10,030 episodes, and report the success rate (%) for each perturbation category. EP: embodied pre-training. **Bold**: best per column; underline: second best.

			Perturbation type							
Method	EP	LIBERO	Cam.	Robot	Lang.	Light	Bkgnd.	Noise	Layout	Avg.
With embodied pre-training										
π0.5	✓	96.9	75.4	77.5	85.6	96.9	94.6	89.7	**85.7**	**85.7**
GE-ACT	✓	96.5	60.7	77.0	77.4	95.8	86.0	90.9	80.2	80.3
VLA-JEPA	✓	97.2	63.3	67.1	85.4	95.6	93.6	66.3	85.1	79.5
X-VLA	✓	98.1	23.4	**89.7**	75.7	88.2	**96.0**	62.7	71.8	71.4
OpenVLA-OFT	✓	97.1	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π0-Fast	✓	85.5	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
π0	✓	94.2	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
UniVLA	✓	95.2	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
Without embodied pre-training										
Faster-WAM	–	98.0	67.9	49.0	92.1	94.3	57.0	82.3	82.7	75.0
4D-WAM	–	98.4	45.2	64.3	90.6	94.3	57.7	69.1	79.2	71.0
Fast-WAM-Joint	–	98.5	42.5	60.8	**92.9**	91.8	55.4	63.3	78.4	68.8
WorldVLA	–	79.1	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
Fast-WAM	–	97.6	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
Ours	–	**98.7**	61.0	79.0	92.0	**97.8**	64.7	78.6	83.6	79.4

Benchmarks and baselines. We conduct experiments on LIBERO (Liu et al., 2023) and LIBERO-PLUS (Fei et al., 2025). LIBERO comprises four 10-task suites (OBJECT, SPATIAL, LIBERO-10, and GOAL), while LIBERO-PLUS extends these tasks to 10,030 perturbed evaluation instances, enabling systematic evaluation under distribution shifts. We further evaluate on ROBOTWIN 2.0 (Chen et al., 2025), a dual-arm manipulation benchmark with clean and domain-randomized settings, to assess whether the findings transfer across benchmarks and embodiments.

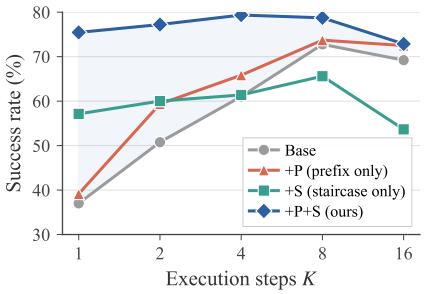


Figure 3: Execution-horizon sweep on LIBERO-PLUS.

Finally, we conduct experiments on both static and dynamic real-world manipulation tasks using a Franka Research 3 arm equipped with a Robotiq 2F-85 gripper, two external Intel RealSense D435i cameras, and a wrist-mounted ZED Mini camera (Section 4.3). We compare StreamingWAM against 13 competitive baselines spanning both VLA policies and WAMs, grouped according to whether they use embodied data during pretraining. Detailed baseline descriptions are provided in the supplementary material.

Implementation details Following Fast-WAM (Yuan et al., 2026), we build our WAM on Wan2.2-TI2V-5B. The action-expert backbone is initialized by linearly interpolating the pretrained video-DiT weights to its dimensions and applying α-scaling to preserve their magnitude, while the action-specific encoder and output head are randomly initialized. At inference, $\mathcal{B}_0$ is initialized using a 10-step flow-matching sampler and re-noised according to the staircase. Each subsequent planning call uses a single Euler update. Detailed training and inference configurations, including the number of optimization and denoising steps, action and execution horizons, are provided in Section A.

4.2 MAIN RESULTS

Table 1 shows that StreamingWAM achieves the strongest overall performance among WAMs without embodied-data pretraining, reaching 98.7% on LIBERO and 79.4% on LIBERO-PLUS. The gain is

most pronounced under distribution shift, where it outperforms the matched Fast-WAM-Joint baseline by 10.6 percentage points on LIBERO-Plus, particularly under camera-viewpoint and robot-initial-state perturbations.

For Robotwin 2.0, as shown in Table 2, all methods are trained only on the same 2,500 clean demonstrations. StreamingWAM leads WAMs without embodied-data pretraining in both settings, reaching 84.2% in the clean setting and 5.1% with randomized object poses. The remaining gap to embodied-pretrained $\pi_{0.5}$ highlights the importance of large-scale cross-embodiment data for out-of-distribution generalization.

Inference cost We report end-to-end time to first action (TTFA), from the availability of a new observation to the first executable action. In simulation (Figure 4a) and on the real robot (Figure 4(c)), StreamingWAM achieves both the lowest TTFA and the highest success rate, with steady-state TTFA of 72.9 ms and 57.3 ms, respectively; relative to Fast-WAM-Joint, these are $5.2\times$ and $6.4\times$ speedups. The one-time warm-start latency is reported separately and incurred only at episode initialization.

Action continuity StreamingWAM produces smoother actions in both simulation and real-world execution. On LIBERO-Plus, the boundary-to-interior jump ratio drops from 1.840 to 1.105 and global translation-command jerk decreases by 59.0% relative to Fast-WAM-Joint (Figure 4b and Table 8).

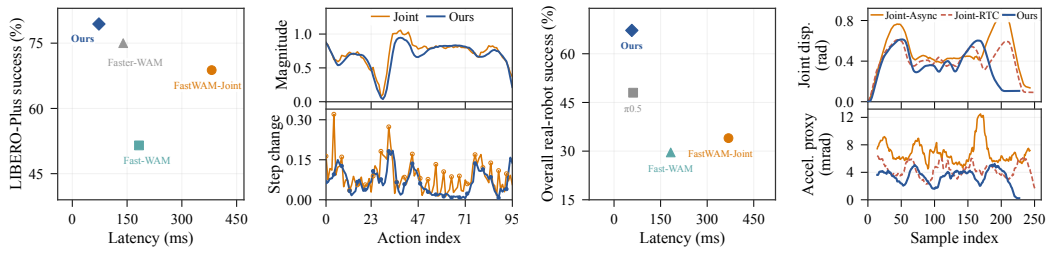
Table 2: Success rate (%) on RoboTwin 2.0. EP: embodied pre-training. **Bold**: best per column; underline: second best.

Method	EP	Clean	Random	Avg.
With embodied pre-training				
Spatial Forcing	✓	77.2	9.5	43.4
π0.5	✓	70.7	**46.0**	**58.4**
X-WAM	✓	70.0	<u>25.8</u>	<u>47.9</u>
X-VLA	✓	68.0	20.9	44.5
EventVLA	✓	65.6	15.7	40.7
Xiaomi-Robotics-0	✓	62.9	18.2	40.6
G0	✓	62.7	9.1	35.9
Abot-M0	✓	57.4	22.9	40.2
Without embodied pre-training				
Fast-WAM-Joint	–	<u>80.3</u>	3.9	42.1
Fast-WAM	–	77.8	1.9	39.9
AHA-WAM	–	64.3	3.2	33.8
Ours	–	**84.2**	5.1	44.7

Real-robot trajectories show the same trend, with smoother joint displacement and a lower discrete-acceleration proxy (Figure 4(d)).

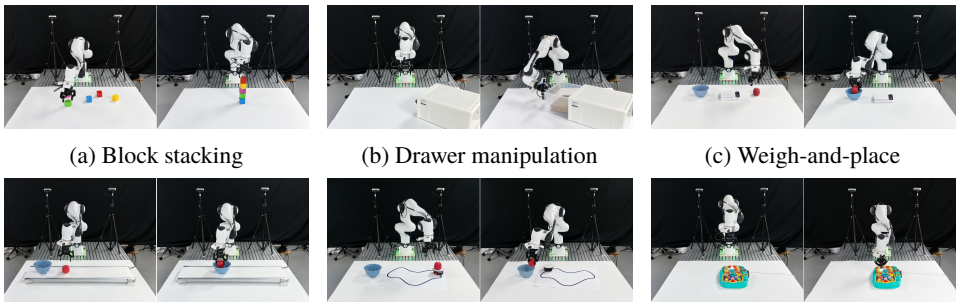
4.3 Real robot experiments

Setup We deploy on a Franka Research 3 arm with a Robotiq 2F-85 gripper, two external Intel RealSense D435i cameras, and a wrist-mounted ZED Mini following the DROID (Khazatsky et al., 2024) configuration. The control loop runs at 15 Hz on an RTX 5090 workstation. Demonstrations are collected via GELLO joint-space teleoperation; details are provided in Section E. On the deployment GPU, steady-state per-replan latency is 57.3 ms for our model (1 denoise step) and 61.5 ms for $\pi_{0.5}$ (10 steps), both within the 66.7 ms control budget. Fast-WAM and FastWAM-Joint require 182.6 ms and 368.6 ms, respectively (10 steps), missing multiple deadlines per replan (Table 7). The six evaluation tasks are illustrated in Figure 5, with detailed descriptions provided in Section E.1.



(a) Simulation efficiency. (b) Simulation continuity. (c) Real-robot efficiency. (d) Real-robot continuity.

Figure 4: Inference efficiency and action continuity in simulation and on the real robot.



(a) Block stacking (b) Drawer manipulation (c) Weigh-and-place

(d) Conveyor handover (e) Car pursuit (f) Whack-a-mole

Figure 5: Real-robot task gallery. Each panel shows two keyframes from a single successful episode. (a)–(c) Static tasks; (d)–(f) dynamic tasks requiring real-time reaction to object motion.

Results Figure 6 shows that StreamingWAM performs best overall. It achieves the highest aggregate success across the five manipulation tasks (168/250 vs. 120/250 for the strongest baseline), leads both dynamic tasks (32/50 on conveyor handover and 20/50 on car pursuit, compared with baseline maxima of 15/50 and 8/50), and records the most whack-a-mole presses (424 vs. 408). These gains suggest that smoother cross-chunk execution improves reliability on static tasks, while anticipatory prediction and rapid response improve success on dynamic tasks.

4.4 Ablation studies and analysis

In this section, we ablate the key components of StreamingWAM and conduct further analyses of its behavior, yielding the following findings:

(1) The staircase flattens the performance curve across execution horizons, and the prefix raises its overall level. Figure 3 and Table 3 isolates the clean prefix (P) and streaming staircase (S) across execution horizons. The staircase reduces Spread from 35.75 pp for Base and 34.57 pp for +P to 11.95 pp for +S; combining P and S further reduces it to 6.50 pp while achieving the highest success at every evaluated K. Thus S provides K-robustness, while P raises the performance level, making the two components complementary.

(2) Multi-width staircase training improves performance across execution horizons. As shown in Table 3, replacing the full training set $\mathcal{K}$ with a fixed staircase width of $K = 4$ preserves low variation across execution horizons but reduces success at every evaluated value of K. Moreover, the fixed-width model does not achieve its best performance at the training width, indicating limited spe-

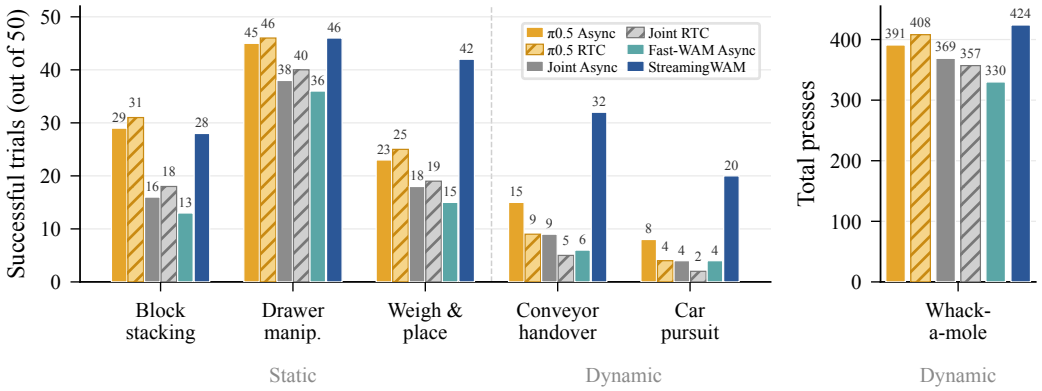


Figure 6: Real-robot results. Each of the five manipulation tasks (left) reports the number of successful trials out of 50; whack-a-mole (right) reports total button presses across 50×30 s sessions.

Table 3: Ablation studies on LIBERO-PLUS (success rate, %). The top block isolates the clean action prefix with dense visual history (P) and the streaming staircase (S); the bottom block varies one design choice from the full model while keeping both P and S. Spread is max−min across all K.

Variant	K=1	2	4	8	16	Spread
Base (Fast-WAM-Joint)	37.01	50.77	61.03	72.76	68.78	35.75
+ Prefix (P)	39.14	59.36	65.83	73.71	72.49	34.57
+ Staircase (S)	57.13	60.01	61.39	65.60	53.65	11.95
+ P + S (full model)	**75.45**	**77.22**	**79.36**	**78.73**	**72.86**	**6.50**
w/ single-width K=4	71.52	74.08	75.31	73.74	70.39	4.92
w/ sparse history	56.35	67.80	69.86	68.45	60.43	13.51
w/ uniform loss weight	75.52	78.37	78.11	77.50	71.27	7.10
w/o warm start	70.30	72.86	74.21	75.23	72.56	4.93

cialization to that particular schedule. These results support the design motivation in Section 3.2: sampling multiple widths exposes the model to more diverse staircase geometries and improves performance across the entire K-sweep.

(3) Dense visual history improves success rates and reduces variation across execution horizons. With P and S held fixed, replacing dense visual history (stride 1) with sparse history (stride 4) shifts the entire K-sweep downward and nearly doubles the Spread (Table 3). The dense frames supply recent visual context that overlaps with the clean-prefix window, giving the model a more accurate estimate of the current scene state. The benefit is most pronounced at small K, where frequent replanning places greater importance on recent observations.

(4) Auxiliary training choices mainly affect accuracy rather than horizon robustness. Uniform loss weighting stays within 1.59 pp of the full model across all K and changes Spread only from 6.50 to 7.10 pp, indicating that position-dependent weighting is a modest refinement. Removing the warm start reduces success at every K, including a 5.15 pp drop at $K = 4$, while leaving Spread comparable (4.93 vs. 6.50 pp). Thus, neither choice is the primary source of K-robustness: the staircase schedule provides robustness, while the warm start mainly raises the overall accuracy level.

(5) The streaming advantage vanishes when the execution width spans the full prediction horizon. Because training uses $\mathcal{K} = \{1, 2, 4, 8\}$, $K = 16$ is an extrapolation setting. At $K = H = 16$, the horizon becomes one block and loses the inter-block staggering required for streaming. The full model then drops by 5.87 pp from $K = 8$ to 16 (78.73% to 72.86%), while its margin over the prefix-only variant shrinks from 13.53 pp at $K = 4$ to 0.37 pp at $K = 16$. Thus, this boundary stress test shows that the streaming benefit vanishes when the execution width spans the prediction horizon.

5 LIMITATIONS

Our study has two main limitations. First, all experiments use a MoT-style dual-expert architecture, leaving generalization to other WAM designs for future work. Second, StreamingWAM lacks large-scale embodied pre-training, limiting out-of-distribution generalization relative to models such as $\pi_{0.5}$. Combining streaming denoising with embodied pre-training is a promising next step.

6 CONCLUSION

We presented StreamingWAM, a streaming-diffusion framework that converts a trained world action model into a single-step rolling policy. By introducing a noise ladder, a prefix rollout module, and dense video-history memory, StreamingWAM reduces steady-state inference latency to 72.9 ms—a $5\times$ reduction over the base model—while improving action continuity across replanning boundaries. Experiments on LIBERO-Plus, RoboTwin 2.0, and real-world robotic tasks show that these gains translate into competitive success rates across both static and dynamic settings, with the execution-horizon sensitivity compressed from 35.5 to 6.5 percentage points.

AI USE STATEMENT

Generative AI tools were used to improve language readability and assist with code editing. All AI-assisted outputs were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Implementation details, training configurations, and additional experimental results are provided in the appendix. Anonymous code and model configurations are included in the supplementary material.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL http://arxiv.org/abs/2410.24164.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025. URL http://arxiv.org/abs/2512.05964.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions, 2025. URL http://arxiv.org/abs/2505.06111.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. Aha-wam:asynchronous horizon-adaptive world-action modeling with observation-guided context routing. abs/2606.09811, 2026. doi: 10.48550/ARXIV.2606.09811. URL https://doi.org/10.48550/arXiv.2606.09811.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model, 2025. URL http://arxiv.org/abs/2506.21539.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. URL http://arxiv.org/abs/2407.01392.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL http://arxiv.org/abs/2506.18088.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. URL http://arxiv.org/abs/2303.04137.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models, 2025. URL http://arxiv.org/abs/2510.13626.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising, 2026. URL http://arxiv.org/abs/2604.26694.

Sigmund H. Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models, 2024. URL http://arxiv.org/abs/2406.04806.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024. URL http://arxiv.org/abs/2403.12945.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL http://arxiv.org/abs/2502.19645.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning, 2026. URL http://arxiv.org/abs/2601.16163.

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, Meng Guo, and Shanghang Zhang. Efficient-WAM: A 1B-parameter world-action model with low-cost future imagination, 2026a. URL http://arxiv.org/abs/2606.10040.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026b. URL http://arxiv.org/abs/2601.21998.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie envisioner: A unified world foundation platform for robotic manipulation, 2025. URL http://arxiv.org/abs/2508.05635.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL http://arxiv.org/abs/2306.03310.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. FASTER: rethinking real-time flow vlas. abs/2603.19199, 2026. doi: 10.48550/ARXIV.2603.19199. URL https://doi.org/10.48550/arXiv.2603.19199.

Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-wam: Do world action models need deep action modules?, 2026. URL http://arxiv.org/abs/2608.02365.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models, 2025. URL http://arxiv.org/abs/2501.09747.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL http://arxiv.org/abs/2504.16054.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model, 2026. URL http://arxiv.org/abs/2602.10098.

MotuBrain Team. World action models in real time: An empirical study of smooth execution via asynchronous deployment. *CoRR*, abs/2608.01880, 2026. doi: 10.48550/ARXIV.2608.01880. URL https://doi.org/10.48550/arXiv.2608.01880.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied AI. abs/2605.12090, 2026a. doi: 10.48550/ARXIV.2605.12090. URL https://doi.org/10.48550/arXiv.2605.12090.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, Yu-Kai Wang, Yuhan Rui, Shenzhe Yao, Zhen Yuan, Zhenhao Shen, Kefei Zhu, Zijie Zhu, Ning Gao, Xiaowei Chi, Guanqi He, Shanghang Zhang, Hao Dong, Lin Shao, and Hang Zhao. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv: 2609.07398*, 2026b.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, et al. 4D-WAM: Infusing spatiotemporal awareness into world action models through trajectory fields, 2026. URL http://arxiv.org/abs/2608.08023.

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, 2026a. URL http://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies, 2026b. URL http://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL http://arxiv.org/abs/2603.16666.

Zhanguang Zhang, Zhiyuan Li, Behnam Rahmati, Rui Heng Yang, Yintao Ma, Amir Rasouli, Sajjad Pakdamansavoji, Yangzheng Wu, Lingfeng Zhang, Tongtong Cao, Feng Wen, Xinyu Wang, Xingyue Quan, and Yingxue Zhang. Do world action models generalize better than vlas? A robustness study. *CoRR*, abs/2603.22078, 2026. doi: 10.48550/ARXIV.2603.22078. URL https://doi.org/10.48550/arXiv.2603.22078.

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

A Reproducibility details

A.1 Model configuration

We use the same dual-expert Mixture-of-Transformers backbone in all three domains. The Video and Action DiTs contain 30 aligned blocks. In every block, the two experts form their own queries, keys, and values, perform joint mixed self-attention, and then return to expert-specific cross-attention and feed-forward paths. The Action DiT residual width is 1,024, whereas its 24 heads use an internal QKV width of $24 \times 128 = 3{,}072$; these are different quantities.

Table 4: Core model configuration by benchmark. Video/Action values are shown as V/A.

Benchmark	Layers	Hidden dim.	FFN dim.	Heads × dim.	Act./prop.	Cameras	Resolution
LIBERO / LIBERO-Plus	30/30	3,072/1,024	14,336/4,096	24×128	7/8	2, horiz.	224×448
RoboTwin 2.0	30/30	3,072/1,024	14,336/4,096	24×128	14/14	3, top + 2 bottom	384×320
Real robot	30/30	3,072/1,024	14,336/4,096	24×128	8/8	3, vertical	384×224

A.2 Training configuration

Within each benchmark, Fast-WAM-Joint and StreamingWAM use the same data pipeline, optimizer, learning-rate schedule, precision, and batch construction; the stages and budgets are listed explicitly in Table 5. All of our runs use AdamW with learning rate 10^{-4}, betas $(0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-2}, gradient clipping at 1.0, and bf16 precision. The learning rate uses 5% linear warm-up followed by cosine decay to 10^{-6}. For StreamingWAM, 80% of samples use the staircase objective with $\mathcal{K}_{\text{train}} = \{1, 2, 4, 8\}$, 32 noise levels, and a random staircase offset; the remaining 20% use ordinary shared-timestep flow matching. The video and action objectives have equal weight, the clean 16-step action prefix is conditioning only, and the future horizon is $H = 16$. During training, we use execution horizons K that evenly divide the action horizon H. We additionally evaluate non-divisor values, such as $K = 10$ when $H = 16$. Although the final incomplete action block may retain a small amount of residual noise, this does not impair deployment-time generalization: the success rate remains consistent with the overall K-sweep trend and exhibits no abrupt degradation.

Table 5: Training schedule by benchmark. For RoboTwin 2.0 and the real robot, the StreamingWAM step count denotes the additional streaming stage after weights-only initialization from the corresponding Fast-WAM-Joint checkpoint.

Method	Scheduler	Steps	Global batch size
LIBERO / LIBERO-Plus			
Fast-WAM-Joint	5% linear warm-up + cosine	83,400	64
StreamingWAM	5% linear warm-up + cosine	83,400	64
RoboTwin 2.0			
Fast-WAM-Joint	5% linear warm-up + cosine	135,776	64
StreamingWAM	5% linear warm-up + cosine	135,776	64
Real robot			
Fast-WAM-Joint	5% linear warm-up + cosine	20,000	128
StreamingWAM	5% linear warm-up + cosine	30,000	128

The FastWAM-Joint rows use ordinary shared-timestep flow matching, whereas the StreamingWAM rows use the 80/20 mixture above. All other methods retain the training values reported in their original papers—including their reported optimization budget, batch size, and learning-rate settings—rather than being silently retrained with our recipe. In particular, Fast-WAM follows Yuan et al. (2026), Faster-WAM follows Ma et al. (2026), and the VLA baselines follow their cited original implementations. We do not infer an unreported hyperparameter.

A.3 Evaluation and inference configuration

LIBERO-Plus results use the full 10,030-episode protocol (object 2,518 / spatial 2,402 / libero-10 2,519 / goal 2,591). Success is determined by the environment's built-in criterion with no tolerance

Table 6: Inference hyperparameters used in our evaluations. H is the predicted future-action horizon (the 16 clean prefix actions of StreamingWAM are excluded), and K is the number of actions executed per replan. $10 \rightarrow 1$ denotes a one-time 10-step warm start followed by one rolling SDP update per replan. For real-robot non-streaming rows, $10/1$ means static/dynamic tasks.

	LIBERO / LIBERO-Plus			RoboTwin 2.0			Real robot		
Method	Steps	H	K	Steps	H	K	Steps	H	K
$\pi_{0.5}$ Async	10	10	5	10	50	50	10	16	4
$\pi_{0.5}$ RTC	10	10	5	10	50	50	10	16	4
Fast-WAM	10	32	10	10	32	24	$10/1$	32	4
Faster-WAM	10	32	4	10	32	4	10	32	4
Fast-WAM-Joint Async	10	32	10	10	32	24	$10/1$	32	4
Fast-WAM-Joint RTC v2	10	32	4	10	32	4	10	32	4
StreamingWAM	$10 \rightarrow 1$	16	4	$10 \rightarrow 1$	16	8	$10 \rightarrow 1$	16	4

modification. RoboTwin 2.0 results use all 50 tasks with seed 0 under both clean and randomised initialisations, following the official leaderboard protocol. Real-robot demonstrations are collected with a GELLO joint-level teleoperation device at 15 Hz (100 per task; 1,028 training trajectories across seven prompts). Evaluation uses 50 trials per manipulation task and 50×30 s sessions for whack-a-mole. Within $\pi_{0.5}$ and FastWAM-Joint, Async and RTC variants share a checkpoint and differ only in execution strategy.

Introduced by Lu et al. (2026), TTFA measures the wall-clock time from a new observation to the first executable action at the controller. In synchronous simulation, TTFA is the end-to-end per-call latency. For StreamingWAM, we report steady-state TTFA after buffer initialization and report the one-time episode-level warm-up separately. All latency measurements use the same RTX 5090 GPU with CUDA-synchronous timing.

During simulation evaluation, StreamingWAM uses synchronous control: the environment waits for each policy call before executing the returned K-action block. Real-robot deployment instead uses asynchronous control. For $K = 4$, the next inference is launched only after the first three actions of the current block have finished, so inference overlaps with execution of the fourth action.

The $\pi_{0.5}$ simulation entries use the public implementation defaults: LIBERO predicts 10 actions and executes five per query, while RoboTwin predicts and executes 50; both use 10 flow-matching integration steps. Async and RTC therefore share the same simulation defaults and differ only in the real-robot execution stack.

The RoboTwin Joint setting executes $K = 24$ actions, whereas StreamingWAM uses $K = 8$; the latter value is an execution setting and must not be confused with the training set $\mathcal{K}_{\text{train}}$. The real-robot FastWAM-Joint RTC v2 row is a validated shadow configuration, not a reported task-success result. StreamingWAM's steady-state one is a rolling update of a persistent buffer, not one-step denoising of an independently redrawn Joint chunk.

A.4 Baselines

We compare StreamingWAM against 13 competitive baselines spanning both VLA policies and WAMs, grouped according to whether they use embodied data during pretraining. Eight methods use embodied-data pretraining: π0.5 (Physical Intelligence et al., 2025), GE-ACT (Liao et al., 2025), VLA-JEPA (Sun et al., 2026), X-VLA (Zheng et al., 2025), OpenVLA-OFT (Kim et al., 2025), π0-Fast (Pertsch et al., 2025), π0 (Black et al., 2024), and UniVLA (Bu et al., 2025). The remaining five do not: Faster-WAM (Ma et al., 2026), 4D-WAM (Yang et al., 2026), WorldVLA (Cen et al., 2025), AHA-WAM (Cai et al., 2026), and Fast-WAM (Yuan et al., 2026). Among these, Fast-WAM-Joint, the jointly trained variant of Fast-WAM, serves as our closest direct baseline because it shares the same model architecture with StreamingWAM while omitting the three modules introduced in Section 3.2.

B Extended simulation results

B.1 Inference latency

Table 7 lists the steady-state per-replan latency for all open-source WAM variants measured on the same RTX 5090 GPU with CUDA synchronous timing. Baseline architectures are reproduced from the published code with random weights (network structure determines latency, not learned weights); our model uses the exact checkpoint reported in the paper.

Table 7: Inference latency comparison. All measurements on the same RTX 5090 with CUDA synchronous timing. n: number of timed forward passes.

Method	Denoising steps	n	Latency (ms)	LIBERO-Plus (%)
Fast-WAM-Joint	10	30	381.6 ± 6.9	68.8
Fast-WAM	10	30	182.6 ± 3.2	51.5
Faster-WAM	10	30	139.4 ± 1.7	75.0
Ours (steady)	1	300	$\mathbf{72.9} \pm 0.3$	**79.4**
Ours (warm-start)	10	30	406.2 ± 0.8	—
Real-robot deployment (same RTX 5090)				
Fast-WAM-Joint	10	30	368.6 ± 2.4	—
Fast-WAM-Joint (timing only)	1	30	46.8 ± 0.3	—
Fast-WAM	10	30	182.6 ± 3.2	—
$\pi_{0.5}$	10	30	61.5 ± 1.2	—
Ours (steady)	1	300	$\mathbf{57.3} \pm 0.6$	—
Ours (warm-start)	10	30	436.8 ± 1.6	—

The 1-step FastWAM-Joint timing row uses the same checkpoint as the 10-step row. In the real-robot task results, FastWAM-Joint uses 10 steps on static tasks and one step in the dynamic-task main setting, as summarized in Table 6.

B.2 Action continuity

Table 8 reports the action-continuity metrics underlying Figure 4b. All three models are evaluated with 1-step inference and $K{=}4$ on a 500-episode balanced subset of LIBERO-Plus. Translation and rotation jumps are the ℓ_2 norm of the end-effector command change at replanning boundaries; the boundary-to-interior ratio measures the ratio of the jump at step 0 to the average jump at non-boundary steps; global translation jerk is the mean ℓ_2 norm of the finite- differenced acceleration of end-effector translation commands.

Table 8: Action-continuity metrics on LIBERO-Plus (500-episode subset, 1-step inference, $K{=}4$). $\downarrow$: lower is better.

Method	SR (%)	Trans. jump $\downarrow$	Rot. jump $\downarrow$	Bnd./int. ratio $\downarrow$	Jerk $\downarrow$
Fast-WAM	60.2	0.1350	0.02903	1.831	5.775
Fast-WAM-Joint	68.8	0.1379	0.02681	1.840	5.710
Ours	**79.0**	**0.0684**	**0.00879**	**1.105**	**2.341**
Ours vs. Fast-WAM-Joint		−50.4%	−67.2%	—	−59.0%

B.3 Per-suite breakdown on LIBERO-Plus

Table 9 breaks down the success rate by suite and perturbation type for Fast-WAM-Joint (Joint) and our model. Our model improves on every suite, with the largest gains on object (+16.4 pp) and libero-10 (+13.8 pp). Across perturbation types, the gap is widest under camera-viewpoint (+18.5 pp pooled) and robot-initial-state (+18.2 pp) perturbations, where Fast-WAM-Joint is most vulnerable.

Table 9: Per-suite per-perturbation success rate (%) on LIBERO-PLUS (K=4, 10,030 episodes). Joint = Fast-WAM-Joint. BG: background textures; Robot: robot initial states; Cam: camera viewpoints; Lang: language instructions; Noise: sensor noise; Obj: objects layout; Light: light conditions.

Suite		BG	Robot	Cam	Lang	Noise	Obj	Light	All
OBJECT	Joint	64.1	57.8	49.2	97.7	68.0	77.4	97.3	72.2
	Ours	83.9	82.2	83.6	97.7	92.2	83.9	98.0	**88.6**
SPATIAL	Joint	74.0	66.3	55.6	93.3	72.1	90.4	99.0	78.5
	Ours	73.6	83.7	64.4	97.4	87.5	96.4	97.9	**86.1**
LIBERO-10	Joint	32.2	72.3	30.8	96.6	64.1	87.8	84.7	66.3
	Ours	53.6	88.3	54.7	100.0	76.6	92.9	98.5	**80.1**
GOAL	Joint	54.4	48.2	35.8	84.9	48.8	61.6	85.3	59.0
	Ours	50.9	62.8	42.6	74.4	57.5	64.9	96.8	**63.4**
Pooled	Joint	55.4	60.8	42.5	92.9	63.3	78.4	91.8	68.8
	Ours	64.7	79.0	61.0	92.0	78.6	83.6	97.8	**79.4**

C PER-TASK RESULTS ON ROBOTWIN 2.0

Table 10 reports the per-task success rates of StreamingWAM under the clean and randomized object-pose settings. Averaging the 50 tasks gives 84.2% and 5.1%, respectively, matching the aggregate results in Table 2.

Table 10: Per-task success rate (%) of StreamingWAM on all 50 RoboTwin 2.0 tasks. The table is arranged in two blocks for compactness.

Task	Clean	Randomized	Task	Clean	Randomized
adjust_bottle	100.0	2.0	place_can_basket	60.0	6.0
beat_block_hammer	93.0	1.0	place_cans_plasticbox	96.0	6.0
blocks_ranking_rgb	99.0	2.0	place_container_plate	99.0	4.0
blocks_ranking_size	45.0	0.0	place_dual_shoes	86.0	2.0
click_alarmclock	100.0	40.0	place_empty_cup	91.0	5.0
click_bell	100.0	17.0	place_fan	80.0	0.0
dump_bin_bigbin	93.0	1.0	place_mouse_pad	54.0	0.0
grab_roller	98.0	5.0	place_object_basket	87.0	3.0
handover_block	98.0	0.0	place_object_scale	81.0	0.0
handover_mic	99.0	3.0	place_object_stand	94.0	7.0
hanging_mug	18.0	0.0	place_phone_stand	83.0	0.0
lift_pot	94.0	4.0	place_shoe	94.0	5.0
move_can_pot	85.0	0.0	press_stapler	84.0	3.0
move_pillbottle_pad	95.0	1.0	put_bottles_dustbin	91.0	0.0
move_playingcard_away	96.0	0.0	put_object_cabinet	51.0	1.0
move_stapler_pad	42.0	0.0	rotate_qrcode	86.0	0.0
open_laptop	85.0	0.0	scan_object	87.0	4.0
open_microwave	64.0	5.0	shake_bottle	100.0	28.0
pick_diverse_bottles	75.0	9.0	shake_bottle_horizontally	100.0	22.0
pick_dual_bottles	85.0	11.0	stack_blocks_three	95.0	0.0
place_a2b_left	95.0	3.0	stack_blocks_two	99.0	12.0
place_a2b_right	90.0	1.0	stack_bowls_three	83.0	9.0
place_bread_basket	89.0	13.0	stack_bowls_two	90.0	4.0
place_bread_skillet	90.0	7.0	stamp_seal	65.0	0.0
place_burger_fries	92.0	8.0	turn_switch	54.0	1.0
Mean	**84.2**	**5.1**			

D ADDITIONAL ABLATION RESULTS

D.1 COMPONENT CONTRIBUTIONS AT $K = 4$

D.2 WARM-START ABLATION

Section 4.4 reports the aggregate effect of warm-start length. Table 11 gives the per-setting breakdown on the full 10,030-episode evaluation: removing the warm start drops success by 5.2 pp, while a single step already recovers the gap.

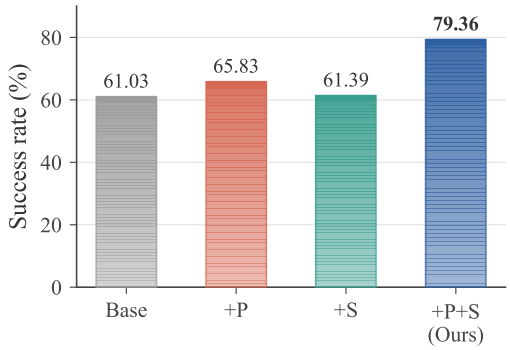


Figure 7: Component contributions on LIBERO-Plus at $K = 4$. The clean action prefix with dense visual history (P) and the streaming staircase (S) each improve over the base and combine additively.

Table 11: Warm-start ablation on LIBERO-Plus at K=4.

Warm-start steps	Successes	Total	Success (%)
0	7,443	10,030	74.2
1	7,934	10,030	79.1
4	7,919	10,030	79.0
8	7,927	10,030	79.0
10	7,960	10,030	79.4

E Real-robot experiment details

E.1 Task descriptions

We evaluate six manipulation tasks, grouped by whether the target object moves during execution. *Static tasks*: **block stacking**—stack five layers of blocks; **drawer manipulation**—grasp the handle and open or close a drawer; and **weigh-and-place**—place an apple on a kitchen scale, read the weight, then transfer it to a bowl. *Dynamic tasks*: **conveyor handover**—pick a moving apple off a conveyor belt and place it into a bowl; **car pursuit**—intercept an apple riding a curved-path toy car and place it into a bowl; and **whack-a-mole**—press the randomly lit button on a response-time toy. The dynamic tasks require the policy to continuously adapt its actions to moving objects and therefore directly evaluate reactive control in time-varying environments.

E.2 Observation preprocessing

Figure 8 shows a representative model input frame from the real robot. Each of the three camera views (two exterior RealSense D435i and one wrist-mounted ZED Mini) is resized to 224×128 and vertically stacked into a 224×384 tensor. In simulation, LIBERO provides two views horizontally concatenated into a 448×224 tensor.

E.3 Action continuity on the real robot

Figure 4(d) compares aligned real-robot trajectory segments from FastWAM-Joint under asynchronous and RTC execution with StreamingWAM. The upper trace reports joint displacement from the initial posture, while the lower trace reports the supplied discrete acceleration proxy. StreamingWAM remains lower than both Joint baselines for most samples; we do not interpret the proxy as physical acceleration or jerk.

E.4 Per-task results

Table 12 reports per-task results for all six real-robot tasks. The five manipulation tasks are evaluated over 50 trials each; whack-a-mole reports total button presses across 50×30 s sessions.

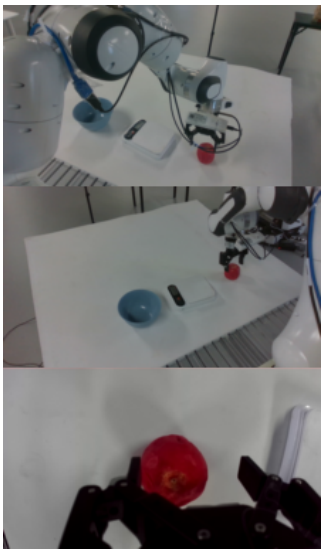


Figure 8: Real-robot model input: three camera views (exterior 1, exterior 2, wrist) vertically concatenated into a 224×384 frame.

Table 12: Real-robot per-task results. Number of successful trials for the five manipulation tasks (50 trials each); total button presses for whack-a-mole (50×30 s sessions).

	$\pi_{0.5}$		FastWAM-Joint		Fast-WAM	**StreamingWAM**
Task	Async	RTC	Async	RTC	Async	**Async**
Static tasks						
Block stacking	29	**31**	16	18	13	28
Drawer manipulation	45	**46**	38	40	36	**46**
Weigh-and-place	23	25	18	19	15	**42**
Dynamic tasks						
Conveyor handover	15	9	9	5	6	**32**
Car pursuit	8	4	4	2	4	**20**
Five-task total	120	115	85	84	74	**168**
Whack-a-mole (presses)	391	408	369	357	330	**424**

E.5 Task execution keyframes

Figures 9 to 14 show successful rollouts of all six real-robot tasks under streaming single-step replanning.

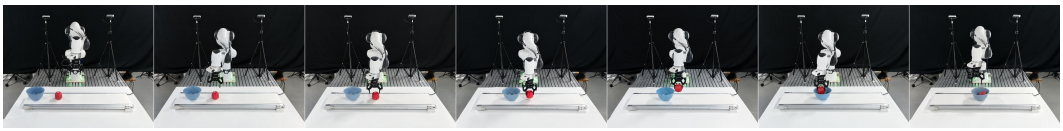


Figure 9: Conveyor-belt handover: the robot picks a moving apple from the belt and places it into the bowl. Seven keyframes from a single successful episode.

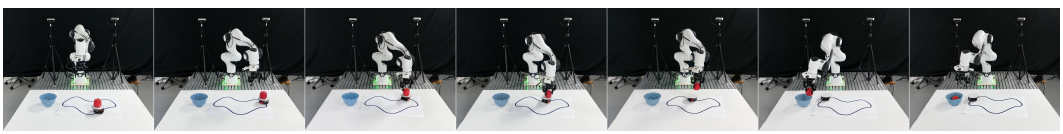


Figure 10: Car pursuit: the robot intercepts an apple riding a curved-path toy car and places it into the bowl. Seven keyframes from a single successful episode.

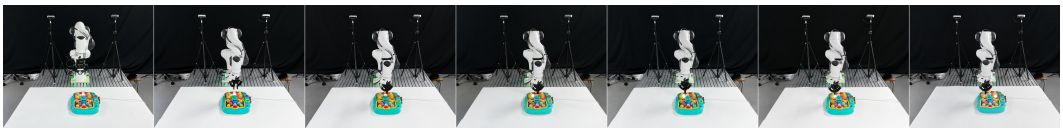


Figure 11: Whack-a-mole: the robot presses the randomly lit button on a response-time toy. Seven keyframes from a single successful episode.

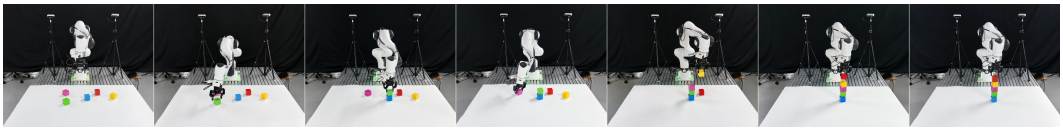


Figure 12: Block stacking: the robot stacks five layers of blocks. Seven keyframes from a single successful episode.

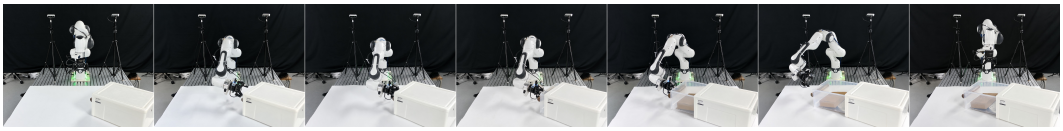


Figure 13: Drawer manipulation: the robot grasps the handle and opens the drawer. Seven keyframes from a single successful episode.

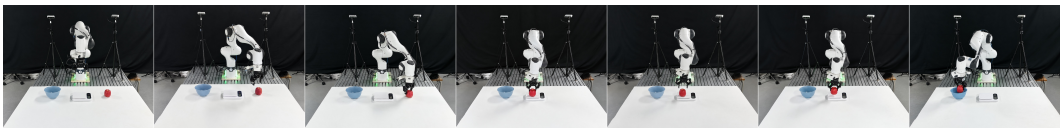


Figure 14: Weigh-and-place: the robot places an apple on a kitchen scale, then transfers it to a bowl. Seven keyframes from a single successful episode.