arXiv:submit/7691189 [cs.RO] 8 Jun 2026

AHA-WAM:Asynchronous Horizon-Adaptive World-Action Modeling with Observation-Guided Context Routing

Jisong Cai[1,2*] **Long Ling**[1,3*] **Shiwei Chu**[1] **Zhongshan Liu**[3]
Jiayue Kang[1] **Zhixuan Liang**[4,2] **Wenjie Xu**[3] **Yinan Mao**[3]
Weinan Zhang[1,2] **Xiaokang Yang**[1] **Ru Ying**[3] **Ran Zheng**[3] **Yao Mu**[1,2†]

[1]Shanghai Jiao Tong University [2]Shanghai AI Laboratory
[3]Baidu AI Cloud [4]The University of Hong Kong

Project Page: https://serene-sivy.github.io/aha-wam/

Abstract: World-action models have emerged as a promising paradigm for robot manipulation, jointly modeling visual scene dynamics and actions to inject physical priors into policy learning. However, existing world-action models couple world prediction and action execution at the same temporal resolution, forcing the world branch to model near-term frame variations that are redundant and weakly informative. We posit that strictly binding world prediction and action execution to the same temporal rhythm may underutilize the potential of the video branch for embodied control. Therefore, we propose AHA-WAM, an **A**synchronous **H**orizon-**A**daptive World-Action Model built on a dual Diffusion Transformer (DiT) architecture that reorganizes world-action modeling around this temporal asymmetry. AHA-WAM instantiates the video DiT as a low-frequency world planner that maintains rolling key-value memory over past observations and exposes reusable layerwise latent context encoding long-horizon scene evolution, while a high-frequency action DiT executes short action chunks in closed loop by querying this context through layerwise joint attention. To support asynchronous execution, we introduce horizon-adaptive offset training and Observation-Guided Video-Context Routing (OVCR), which together let the action expert exploit long-horizon world context while remaining responsive to real-time execution state without rerunning the video DiT. Experiments on RoboTwin and real-world manipulation tasks show that AHA-WAM achieves state-of-the-art performance without any robot-data pretraining, attaining **92.80%** average success on RoboTwin and **78.3%** success across 4 real-world tasks, while reaching **24.17** Hz closed-loop control with a **4.59**$\times$ speedup over Fast-WAM.

Keywords: Robot Learning, Embodied Manipulation, World-Action Model

1 Introduction

Robotic manipulation requires policies that understand not only the current scene, but also how the scene may evolve under the robot's actions. Recent vision-language-action (VLA) models have advanced robot control by scaling imitation learning with large vision-language model backbones, yet action labels provide relatively sparse supervision for the underlying physical dynamics of manipulation. World-action models (WAMs) address this by coupling action prediction with dense video-based world modeling, learning how actions co-evolve with visual scene dynamics to inject physical priors into control and offer a more generalizable, and transferable policy representation.

*Equal contribution.
†Corresponding author. Email: muyao@sjtu.edu.cn.
This work was done during Long Ling's and Jiayue Kang's internship at Shanghai Jiao Tong University.

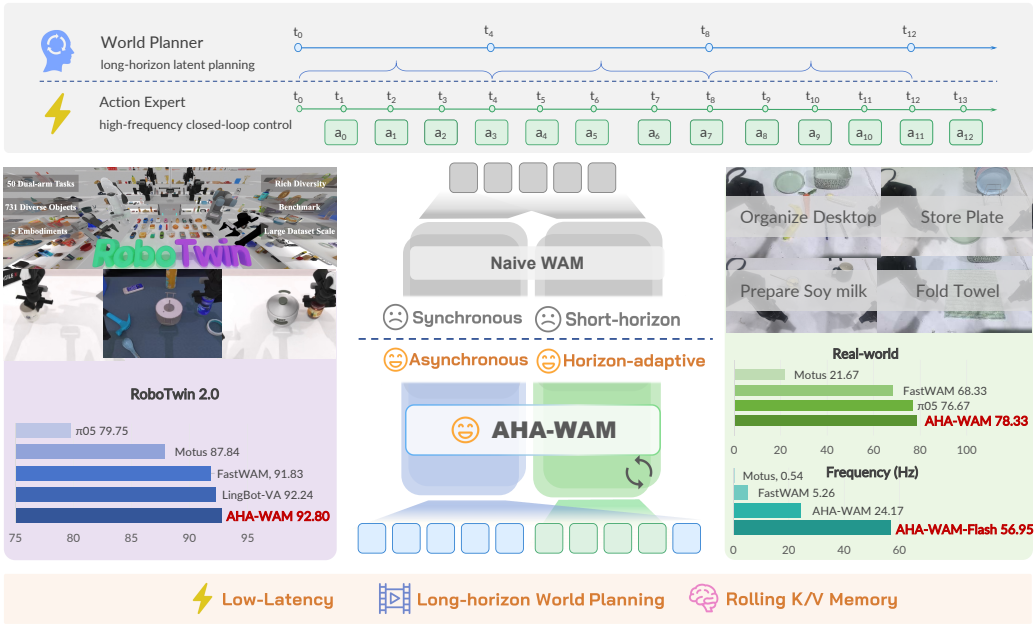

Figure 1: **Overview of AHA-WAM.** AHA-WAM connects past observations, future-oriented world planning, and fast closed-loop action execution: a slow world planner maintains reusable memory and planning context, while a fast action expert adapts that context to the latest observation before predicting short action chunks.

Despite this promise, current WAMs leave an important design space underexplored: how can the video branch more effectively empower robot control? Existing approaches either explicitly roll out future frames for inverse-dynamics decoding or joint world-action modeling, while others reuse the video branch as a latent encoder alongside the action branch. Both designs assume that world prediction and action execution share the same short horizon, which forces the world branch to spend capacity on dense adjacent-frame variations that are often highly correlated and only weakly informative for control. We argue that this coupling reflects a structural mismatch in temporal abstraction: the video world model can better serve embodied control by forming a temporally extended latent plan over future visual states, whereas the action model should remain tightly coupled to the real-time control loop, incorporating the latest observation to issue timely closed-loop corrections.

We propose AHA-WAM, an **A**synchronous **H**orizon-**A**daptive World-Action Model that reorganizes world-action modeling around this temporal asymmetry. AHA-WAM instantiates the video Diffusion Transformer (DiT) as a low-frequency world planner that maintains a rolling key-value (KV) memory over past observations and exposes reusable layerwise latent context encoding long-horizon scene evolution, amortizing expensive world-model computation across multiple action steps. In parallel, a high-frequency action DiT executes short action chunks in closed loop by querying this context through layerwise joint attention. This design preserves the benefits of joint world-action modeling while aligning each branch with the temporal scale at which it's most informative. In AHA-WAM, *horizon-adaptive* refers not to online horizon adjustment, but to a horizon-decoupled formulation where the world and action branches are assigned different temporal horizons according to their functional roles.

A key challenge introduced by asynchronous execution is that the planner context may become stale or misaligned with the current action chunk as the executor runs ahead. To address this, we introduce Observation-Guided Video-Context Routing (OVCR), which constructs chunk-specific latent queries from the latest visual observations to route and update the planner's layerwise video context before it is consumed by the action DiT, giving each action prediction an observation-conditioned view of the slow planner state without feeding dense visual tokens into the high-frequency action branch. Proprioceptive feedback enters the action DiT directly, while visual feedback is injected

indirectly through the routed video context, keeping the reused planner context responsive to real-time execution state without recomputation. We further introduce horizon-adaptive offset training to expose the model to diverse planner-executor phase relationships, and equip the video DiT with rolling KV memory to connect past observations with future plans.

We evaluate AHA-WAM on RoboTwin 2.0 and real-world manipulation tasks. Though without robot-data pretraining, AHA-WAM achieves 92.80% average success across 50 RoboTwin 2.0 tasks, reaching state-of-the-art-level performance among strong VLA and WAM baselines. Across four real-world tasks covering deformable manipulation, long-horizon organization, fine-grained tool use, and spatial generalization, AHA-WAM achieves 78.3% average success, demonstrating robust deployment performance beyond simulation. Across four dimensions of real-world out-of-distribution evaluation, AHA-WAM matches $\pi_{0.5}$ in exhibiting the most limited performance degradation, suggesting stronger robustness to distribution shifts. Benefiting from the asynchronous inference schedule, ODE distillation and cuda optimizations, AHA-WAM reaches up to 56.9 Hz closed-loop control frequency, corresponding to a 10.82$\times$ speedup over Fast-WAM.

In summary, our contributions are threefold:

- First, we propose AHA-WAM, an asynchronous horizon-adaptive world-action model that decouples slow video-DiT world planning from fast action-DiT closed-loop execution, introducing horizon-adaptive offset training to support arbitrary planner-executor phase relationships.
- Second, we develop Observation-Guided Video-Context Routing (OVCR), which dynamically constructs chunk-specific latent video context from current observations to keep asynchronous planner context aligned with real-time execution state.
- Third, we validate AHA-WAM across simulation, real-world manipulation, and latency benchmarks, demonstrating state-of-the-art-level manipulation performance without large-scale robot-data pretraining and substantially improved inference efficiency over existing world-action model baselines.

2 Related Work

Generalist robot manipulation policies. Generalist robot policies aim to learn broadly applicable manipulation skills from large-scale diverse demonstrations [1–4]. RT-1 [5] and RT-2 [6] established the vision-language-action (VLA) paradigm by absorbing heterogeneous robot trajectories and transferring web-scale vision-language knowledge into robotic control. Subsequent works [7–15] further improved generalization by integrating flow-matching and diffusion-based action heads [16, 17] with large pretrained backbones, establishing Diffusion Transformers (DiTs) [18] as expressive and scalable policy architectures. However, these methods remain primarily action-centric, with physical scene dynamics only implicitly captured through demonstrations, which motivates the world-action modeling paradigm that AHA-WAM extends.

World models for robot control. World models [19–22] augment robot policies with predictive visual dynamics beyond action-only imitation. Existing video-based robot world models can be broadly grouped into two lines. The first follows an *imagine-then-act* paradigm: methods such as UniPi [23], Seer [24], and Video Prediction Policy [25] first predict future visual states or predictive visual representations, and then recover actions through inverse dynamics or a policy conditioned on the predicted future, introducing latency in closed-loop deployment. The second line performs *joint world-action modeling*, where future visual dynamics and actions are modeled within a shared generative architecture [26–31]. But all couple world prediction and action execution at the same temporal resolution, spending capacity on short-horizon adjacent frames that are redundant for control. AHA-WAM addresses this by decoupling the video DiT as a low-frequency long-horizon planner from the action DiT as a high-frequency closed-loop executor.

Dual-system and asynchronous robot policies. Recent robot policies have explored dual-system or asynchronous designs to combine deliberative computation with reactive control. RoboDual [32] uses a VLA generalist to provide task understanding and coarse action guidance for an efficient dif-

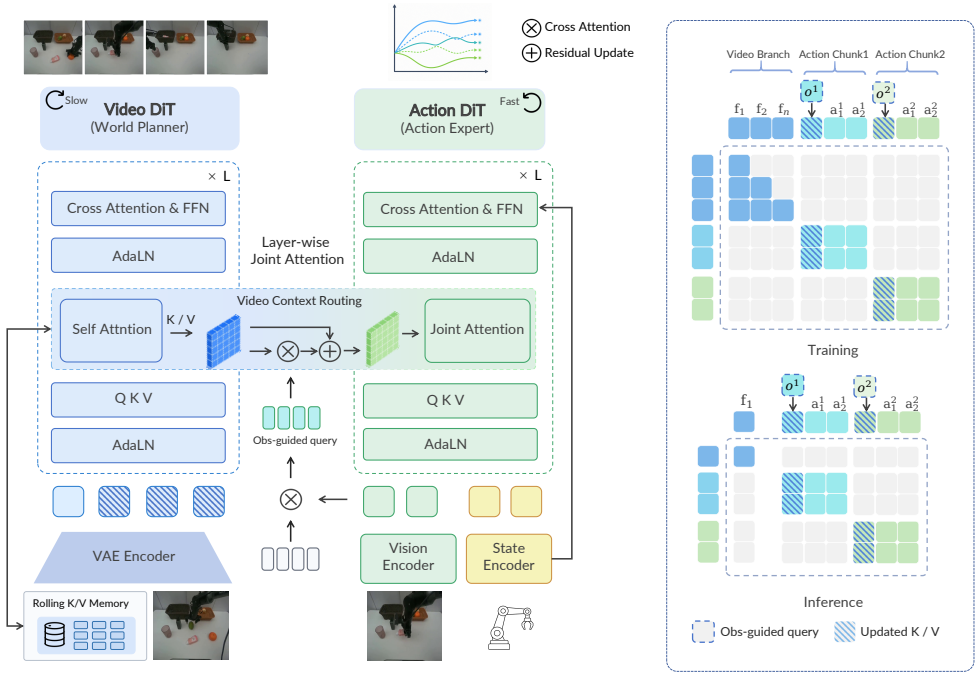

Figure 2: **AHA-WAM architecture and attention mask.** AHA-WAM decouples world planning and action execution into a slow video-DiT planner and a fast action-DiT executor. The video branch is trained with a fully causal mask to learn temporal dynamics. For each action update, the latest observation o^n first queries and updates the video DiT's K/V states through OVCR, producing updated planner context that is consumed by the action DiT through layerwise joint attention.

fusion specialist, while Reactive Diffusion Policy [33] combines a slow latent diffusion policy with a fast tactile feedback pathway for contact-rich manipulation. AsyncVLA [34] runs a large foundation model asynchronously to provide delayed semantic guidance, which is refined by a lightweight edge adapter for fast execution. While these methods validate the effectiveness of dual-system execution, their slow branch primarily provides guidance, feedback, or representations for the action policy. AHA-WAM instead realizes the slow-fast split within joint world-action modeling: the video DiT and action DiT operate asynchronously yet remain coupled through layerwise joint attention.

3 Asynchronous Horizon-Adaptive World-Action Modeling

World-action models couple action prediction with learned visual dynamics, but existing formulations typically organize the video branch and the action branch under the same short-horizon execution rhythm. We instead formulate world-action modeling as a two-timescale generation problem. Figure 2 illustrates the overall architecture. AHA-WAM reorganizes WAM inference into an asynchronous world-action coupling framework while preserving a dual-DiT architecture: a low-frequency video planner produces reusable long-horizon planner context, and a high-frequency action expert consumes this context for closed-loop action denoising. Here, "horizon-adaptive" denotes robustness to arbitrary planner–executor phase offsets under decoupled temporal horizons, rather than online horizon selection. In other words, AHA-WAM does not dynamically choose the video or action horizon at test time; instead, it trains the executor to consume long-horizon planner context under variable action-start offsets, making the asynchronous interface robust to the phase misalignment induced by streaming deployment.

This design is supported by three complementary mechanisms. Observation-Guided Video-Context Routing (OVCR) adapts the cached planner context to the latest observation without rerunning the video DiT. Horizon-adaptive offset training exposes the action expert to the planner–executor phase shifts induced by asynchronous streaming, while rolling K/V memory extends the video planner's

temporal receptive field over past observations. Finally, real-time inference optimizations further accelerate the high-frequency action stream for closed-loop deployment.

3.1 Dual-DiT Planner–Executor Architecture

Problem setting. We consider language-conditioned visuomotor policy learning from visual observations and proprioceptive robot states. At control time t, the policy receives a visual observation history O_t^v, a proprioceptive state s_t, and a language instruction l, and predicts an executable action chunk $A_t = \{a_t, \dots, a_{t+h_a-1}\}$ of horizon h_a. A standard action-centric policy directly models

$$\pi_\theta(A_t \mid O^v_{\leq t}, s_t, l). \tag{1}$$

World-action models augment this policy with visual dynamics learning by jointly modeling future visual evolution and action generation. Let $Z^v_{t:t+h_v}$ denote future video latents over a visual planning horizon h_v. Rather than updating video modeling and action generation under the same short horizon, AHA-WAM decouples their temporal roles: the video branch models a longer horizon h_v, while the action branch predicts executable chunks of horizon h_a, with $h_a < h_v$.

Model architecture. AHA-WAM is built on a dual-DiT architecture consisting of a video DiT world planner and an action DiT executor. Visual observations are encoded by the pretrained VAE, while language embeddings from the text encoder condition both branches. The action branch is a lightweight action expert DiT with the same layer depth as the video DiT, enabling layerwise interaction between the two branches.

The video DiT takes visual latent tokens as input and is trained to predict future video latents over the longer planning horizon. The action DiT receives noisy action tokens and proprioceptive tokens, and denoises the action chunk under closed-loop robot-state feedback. Visual feedback for the high-frequency action branch is not introduced by naively concatenating dense visual tokens; instead, visual information is mediated through the planner video context and later adapted by OVCR in Section 3.2.

Layerwise planner–executor coupling. The central interface between the two DiTs is the layerwise planner video context. Given the visual observation context and language instruction, the video planner produces

$$\mathcal{C}^p_\tau = \left\{ \left(K^{p,\ell}_\tau, V^{p,\ell}_\tau\right) \right\}_{\ell=1}^{L}, \tag{2}$$

where p denotes the planner branch, τ indexes the latest planner refresh and ℓ indexes the transformer layer. This context is a latent world-plan representation exposed by one video-DiT forward and reused by multiple subsequent action-DiT forwards. It differs from the rolling K/V memory in Section 3.4, which stores historical video states across planner refreshes inside the video planner.

During training, the video branch predicts future video latents under a fully causal video mask, encouraging the video DiT to learn forward scene dynamics while shaping the planner context with visual dynamics supervision. The action branch is masked from attending to future video tokens so that during inference, the future-video prediction path can be removed. Thus, future-video prediction serves as a world-modeling training signal, while planner video context serves as the inference-time interface to the action executor.

At each action update, the raw planner context $\mathcal{C}^p_\tau$ is first adapted by OVCR into a chunk-specific context $\widetilde{\mathcal{C}}^p_t = \{(\widetilde{K}^{p,\ell}_t, \widetilde{V}^{p,\ell}_t)\}_{\ell=1}^{L}$. The action DiT then denoises the current action chunk through layerwise joint attention:

$$\bar{H}^{a,\ell}_t = \mathrm{Attn}\left(Q^{a,\ell}_t, \left[K^{a,\ell}_t; \widetilde{K}^{p,\ell}_t\right], \left[V^{a,\ell}_t; \widetilde{V}^{p,\ell}_t\right]\right), \tag{3}$$

where $Q^{a,\ell}_t$, $K^{a,\ell}_t$, and $V^{a,\ell}_t$ are the action-DiT query, key, and value projections, and $\bar{H}^{a,\ell}_t$ is the planner-conditioned action hidden state. This coupling preserves WAM-style interaction between visual dynamics and action generation, while amortizing expensive video-DiT computation across multiple high-frequency action updates.

Joint world-action training objective. AHA-WAM is trained with a joint flow-matching objective over action chunks and future video latents. For a target variable y, either an action chunk A_t or future video latents $Z^v_{t:t+h_v}$, we sample Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ and a flow time $\rho \in (0, 1)$, and construct the interpolated sample

$$y_\rho = (1 - \rho)y + \rho\epsilon. \tag{4}$$

The model predicts the corresponding velocity field:

$$\mathcal{L}_{\mathrm{FM}}(y) = \mathbb{E}_{y,\epsilon,\rho}\left[\|f_\theta(y_\rho, \rho, O^v_t, s_t, l) - (\epsilon - y)\|_2^2\right]. \tag{5}$$

We instantiate this objective for action generation and video co-training:

$$\mathcal{L}_a = \mathcal{L}_{\mathrm{FM}}(A_t), \qquad \mathcal{L}_v = \mathcal{L}_{\mathrm{FM}}(Z^v_{t:t+h_v}). \tag{6}$$

The final objective is

$$\mathcal{L} = \mathcal{L}_a + \lambda\mathcal{L}_v, \tag{7}$$

where λ balances action learning and video dynamics learning. At deployment, AHA-WAM removes explicit future-frame decoding: the video DiT only refreshes planner video context, and the action DiT uses this context to generate closed-loop action chunks.

3.2 Observation-Guided Video-Context Routing

Asynchronous execution reuses one planner video context for multiple action chunks, which amortizes video-DiT computation but creates a context-alignment problem: before the next planner refresh, the robot state and visual scene may have changed. Observation-Guided Video-Context Routing (OVCR) addresses this by using the latest visual observation to convert the shared planner context into a chunk-specific context. Instead of feeding dense visual tokens into the high-frequency action DiT, OVCR compresses them into a small set of routing queries that select and edit the relevant planner context before action denoising.

For each action chunk at time t, we split the aligned observation context into visual tokens X^v_t and proprioceptive tokens X^s_t. The proprioceptive input is compact and directly tied to the instantaneous robot state, so a lightweight encoder maps it to a state token which is directly provided to the action DiT. Visual feedback is injected indirectly through context routing. Given Q learnable base queries $B \in \mathbb{R}^{Q\times d}$, OVCR constructs observation-guided routing queries by attention pooling over the current visual tokens:

$$Z^q_t = \mathrm{Attn}\left(B, f_v(X^v_t), f_v(X^v_t)\right), \tag{8}$$

where f_v is a lightweight visual projection module. The resulting queries provide compact, observation-conditioned slots for retrieving chunk-relevant information from the planner video context.

Let $\mathcal{C}^p_{\tau(t)} = \{(K^{p,\ell}_{\tau(t)}, V^{p,\ell}_{\tau(t)})\}_{\ell=1}^L$ denote the latest available planner context. For each layer ℓ, OVCR first reads planner features using the routing queries and then predicts residual key–value updates:

$$R^\ell_t = \mathrm{Attn}\left(Z^{q,\ell}_t, K^{p,\ell}_{\tau(t)}, V^{p,\ell}_{\tau(t)}\right), \qquad (\Delta K^{p,\ell}_t, \Delta V^{p,\ell}_t) = g^\ell_\psi\left(R^\ell_t, Z^{q,\ell}_t\right), \tag{9}$$

where g^ℓ_ψ is a lightweight layerwise router. The chunk-specific planner context is produced by a gated residual update:

$$\widetilde{K}^{p,\ell}_t = K^{p,\ell}_{\tau(t)} + \alpha^\ell_t \Delta K^{p,\ell}_t, \qquad \widetilde{V}^{p,\ell}_t = V^{p,\ell}_{\tau(t)} + \alpha^\ell_t \Delta V^{p,\ell}_t, \tag{10}$$

where α^ℓ_t is a learned gate. The adapted context $\widetilde{\mathcal{C}}^p_t = \{(\widetilde{K}^{p,\ell}_t, \widetilde{V}^{p,\ell}_t)\}_{\ell=1}^L$ is then consumed by the action DiT through the layerwise joint attention in Eq. 3.

OVCR turns planner-context reuse from static caching into observation-conditioned retrieval and adaptation. Thus, the video DiT remains outside the per-update critical path, while each action chunk still receives a planner representation aligned with the latest visual evidence.

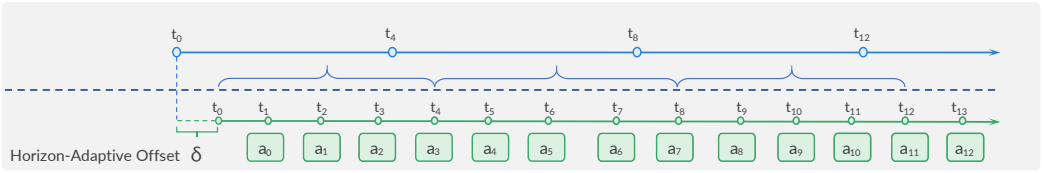

Figure 3: **Horizon-adaptive offset training.** We randomly shift the action-chunk grid by $\delta \in [0, h_a)$ inside the video planning horizon, so the action executor learns to consume planner context under different phase offsets induced by asynchronous deployment.

3.3 Horizon-Adaptive Offset Training

Asynchronous streaming changes the relative temporal phase between the slow planner and the fast executor. If training always uses a fixed alignment between the video planning window and the action chunk, the action DiT may overfit to a single planner–executor phase and become brittle when reusing planner context at intermediate phases during deployment. We therefore introduce horizon-adaptive offset training to expose the executor to the phase shifts induced by asynchronous inference.

Let h_v denote the video planning horizon and h_a denote the action chunk horizon, with $h_a < h_v$. As shown in Figure 3, for each training segment starting at time τ, the video planner models future video latents over $[\tau, \tau + h_v)$. Instead of aligning the first action chunk to the planner start, we sample

$$\delta \sim \mathcal{U}\{0, 1, \ldots, h_a - 1\}, \tag{11}$$

and shift the action-chunk grid by δ within the planner horizon. The action objective is then evaluated over offset-aligned chunks:

$$\mathcal{L}_a = \mathbb{E}_\delta \left[\mathcal{L}_{\mathrm{FM}} \left(A_\tau^\delta \right) \right], \tag{12}$$

where A_τ^δ denotes an action chunk starting from an offset-aligned position inside the planner horizon, and $\mathcal{L}_{\mathrm{FM}}$ is defined in Eq. 5.

Since the planner–executor alignment is periodic with the action chunk size, sampling $\delta \in [0, h_a)$ covers all chunk-level phases encountered when one planner context is reused across multiple executor updates. Thus, horizon-adaptive offset training teaches the action DiT to consume long-horizon planner context under variable action-start offsets, while OVCR handles observation-conditioned context selection at each phase.

3.4 Rolling Planner Memory

Since the video DiT operates as a low-frequency planner, it should preserve historical scene information across planner refreshes rather than relying only on the current observation. This is important for long-horizon manipulation, where completed subgoals, displaced objects, or previously observed states can provide essential context. We therefore maintain a fixed-size FIFO rolling K/V memory inside the video planner.

For each layer ℓ, the memory stores historical video states from recent planner refreshes:

$$M_\tau^\ell = \mathrm{FIFO}\left(M_{\tau-1}^\ell \cup \{(K_\tau^{p,\ell}, V_\tau^{p,\ell})\} \right), \tag{13}$$

where τ indexes planner refreshes and the memory window size is fixed. At the next refresh, the video DiT attends to this memory when producing the new planner video context $\mathcal{C}_\tau^p = \{(K_\tau^{p,\ell}, V_\tau^{p,\ell})\}_{\ell=1}^L$.

This memory is internal to the slow video planner and is not directly consumed by the action DiT. It extends the planner's temporal receptive field before the next planner context is produced, while the high-frequency executor still interacts only with the latest OVCR-adapted planner video context.

3.5 Streaming Inference and Real-Time Optimization

The asynchronous planner-executor architecture naturally leads to a streaming inference schedule. At deployment, the video planner and the action executor are executed as two non-blocking streams, where AHA-WAM removes the video DiT from the per-update critical path, but the action branch must still run at the closed-loop control frequency. We therefore optimize the action-chunk inference path, measuring the end-to-end latency L_{chunk} of one action update, including image encoding, planner-context access, OVCR routing, and action denoising. Video-DiT prefill is executed asynchronously, so L_{chunk} directly determines the action-update frequency reported in Section 4.5.

CUDA acceleration. We compile the repeated action-phase computation into a static deployment path by executing the action DiT, memory and context modules, and VAE encoder through TensorRT and CUDA-graph capture where possible, while removing redundant loop-invariant computations and repeated buffer copies from the denoising hot path. These changes do not alter the model architecture, weights, or sampling procedure, but reduce the 10-step action inference latency from 415.77 ms in PyTorch eager execution to 41.37 ms.

ODE distillation. On top of the CUDA-accelerated 10-step path, we construct AHA-WAM-Flash by distilling the action sampler from 10 denoising steps to 2 steps while keeping the same observation, planner-context, and OVCR interface. The video DiT is frozen during distillation, and the student is trained from sampled intermediate states to directly predict the teacher's final action output, with sampling biased toward noisier states to support aggressive step reduction. ODE distillation further reduces L_{chunk} from 41.37 ms to 17.56 ms. Additional CUDA ablations, distillation schedules, and step-latency tradeoffs are provided in Appendix D.

4 Experiments

We evaluate AHA-WAM in simulation and on real robots to test whether asynchronous world-action modeling preserves manipulation performance while improving closed-loop efficiency. We cover RoboTwin 2.0 [35] performance, component ablations, real-world deployment, generalization analysis and inference latency.

Specifically, our experiments ask five questions including capability, mechanism, deployability, robustness, and efficiency. **Q1.** Can our asynchronous WAM preserve or improve simulation performance while faster? (Sec. 4.2) **Q2.** How important is each component for AHA-WAM? (Sec. 4.3) **Q3.** Can AHA-WAM achieve reliable real-robot performance? (Sec. 4.4) **Q4.** How robust is AHA-WAM under previously unseen real-world task conditions? (Sec. 4.4) **Q5.** How much closed-loop latency reduction does AHA-WAM provide? (Sec. 4.5)

4.1 Experimental Setup

Model and training. AHA-WAM is implemented as a dual-DiT policy with an explicitly asynchronous planner–executor interface. We use the pretrained Wan2.2-5B video model to initialize the world-planning branch, including the video DiT, text encoder, and video VAE. Following Fast-WAM [30], the high-frequency action executor adopts a compact DiT architecture with hidden dimension $d_a = 1024$, corresponding to approximately 1.02B parameters. In addition to the 4.99B-parameter video planner and the 1.02B-parameter action DiT, AHA-WAM introduces 1.22B parameters for rolling K/V memory and context-routing modules, giving a total instantiated model size of about 7.23B parameters.

For the temporal configuration of AHA-WAM, the video branch operates over a long planning horizon of $h_v = 64$, while the action branch predicts short executable chunks with horizon $h_a = 16$. During asynchronous inference, the video planner maintains reusable layerwise K/V context, which is augmented by a FIFO rolling memory over at most 6 historical observation frames. The action executor adapts this context through OVCR, using 32 observation-guided routing queries for each action chunk.

Table 1: **RoboTwin 2.0 average success on 50 tasks.**

Method	Robo. P.T.	Clean (%)	Rand. (%)	Avg. (%)
π_0	✓	65.92	58.40	62.16
$\pi_{0.5}$	✓	82.74	76.76	79.75
ABot-M0	✓	81.20	80.40	80.80
Motus from Wan2.2	✗	77.56	77.00	77.28
Motus	✓	88.66	87.02	87.84
LingBot-VA	✓	92.90	91.50	92.20
Fast-WAM	✗	91.88	91.78	91.83
AHA-WAM-Flash	✗	90.48	89.92	90.20
AHA-WAM	✗	**93.40**	**92.20**	**92.80**

Training uses flow matching for both world modeling and action prediction, with logit-normal sampling of noise times. For action generation, we use 10 denoising steps at inference and set CFG to 1.0. All models are trained with AdamW, a learning rate of 1×10^{-4}, weight decay 0.01, cosine scheduling, mixed precision, and gradient clipping. Latency is reported on a single NVIDIA RTX 5090D GPU. Appendix A lists additional implementation details, including AHA-WAM-Flash distillation and other training configurations.

Evaluation scope. We evaluate AHA-WAM in both RoboTwin 2.0 simulation [35] and real-world experiment. The simulation benchmark tests multi-task manipulation under clean and randomized scenes, while the real-world experiments test deployment on physical bimanual tasks. Across both settings, we report task success rate as the primary metric and additionally analyze closed-loop inference latency to quantify deployment efficiency.

4.2 RoboTwin Simulation

We evaluate AHA-WAM on RoboTwin 2.0 [35], a bimanual manipulation benchmark with 50 dual-arm tasks covering diverse skills. We follow the multi-task training setup of [27, 29, 30]: for each task, the training set contains 50 demonstrations in the clean setting and 500 demonstrations in the randomized setting. Models are trained in a multi-task manner with a global batch size of 512. Each method is tested over 100 trials per task in both the clean and randomized settings, and we report task-averaged success rates. We compare against competitive VLA and WAM baselines, including π_0 [2], $\pi_{0.5}$ [9], ABot-M0 [36], Motus [27], LingBot-VA [29], and Fast-WAM [30]. Appendices B and E provide detailed protocols, baseline settings, and per-task success rates.

Results and analysis. Table 1 summarizes the RoboTwin results: AHA-WAM achieves 93.40% success in the clean setting and 92.20% under randomized evaluation, yielding an average success rate of 92.80%. Without robot data pretraining, AHA-WAM improves over Fast-WAM by 0.97 percentage points, showing that our asynchronous design not only increases the closed-loop control frequency but also preserves and improves performance. AHA-WAM also exceeds LingBot-VA, the strongest baseline with large-scale robot-data pretraining, by 0.60 points. The AHA-WAM-Flash variant retains 90.20% average success, incurring only a modest performance drop while further cutting the latency of AHA-WAM by more than half as analyzed in Section 4.5.

4.3 Ablation Studies on RoboTwin

We ablate the components that make asynchronous world-action modeling effective. Naive-Async decouples the video and action branches, but directly reuses the latest planner context without rolling K/V memory or OVCR. We then add each mechanism separately and jointly.

Table 2: **RoboTwin 2.0 ablations.**

Variant	Clean (%)	Rand. (%)	Avg. (%)
Fast-WAM	91.88	91.78	91.83
Naive-Async	88.64	88.56	88.60
+ KV Memory	91.40	90.62	91.01
+ OVCR	91.52	91.42	91.47
AHA-WAM	**93.40**	**92.20**	**92.80**

Table 2 shows that asynchronous execution alone is insufficient. Naive-Async drops from 91.83% to 88.60%, indicating that faster action up-

dates cannot compensate for stale or phase-misaligned planner context. Adding rolling K/V memory recovers performance to 91.01%, showing that persistent planner states help stabilize the reused video context across refreshes. The gain is moderate, which is expected because most RoboTwin tasks are short-to-medium horizon and keep task-relevant objects visible. OVCR further improves success to 91.47%, suggesting that observation-conditioned context adaptation is the more direct remedy for asynchronous planner–executor mismatch. The full AHA-WAM reaches 92.80%, confirming that memory and routing are complementary: memory preserves temporal context, while OVCR aligns it with the current execution state.

4.4 Real-World Robot Experiments

Setup. We evaluate on a bimanual AgileX Piper platform with an ego-view RGB camera, and policies uses only head-view RGB observations, proprioceptive states and language instructions as input. We consider four tasks—*Fold Towel*, *Organize Desktop*, *Prepare Soy Milk*, and *Store Plate*—covering deformable manipulation, long-horizon rearrangement, fine-grained tool use, contact-rich control, and spatial generalization. For each task, we collect approximately 120 episodes on average. Appendix C provides task details, data scale, execution procedures, and generalization variants.

Implementation. Since Fast-WAM and AHA-WAM are not pretrained on any robot data by default, we pretrain Fast-WAM and AHA-WAM on the selected RoboCOIN subset [37], containing $24{,}600$ trajectories and approximately 165 hours of robot data for a fair and stable deployment comparison. Both models are finetuned on the same task-specific demonstrations. Motus and Fast-WAM incur large naive deployment latency, which leads to sparse action updates and unstable closed-loop behavior. We deploy them with an RTC-style non-blocking execution scheme [38] and action interpolate between predicted chunks, enabling smoother control under high inference latency. AHA-WAM is deployed with its asynchronous planner–executor interface, where the executor performs high-frequency closed-loop updates using the latest observation and reused planner context.

Results. Figure 4 reports real-world performance under original settings and generalization shifts. In the original settings, AHA-WAM achieves 78.33% success, clearly outperforming the WAM baselines Motus (21.67%) and Fast-WAM (68.33%), while matching the strong generalist VLA baseline $\pi_{0.5}$ (76.67%). Under generalization shifts, $\pi_{0.5}$ obtains the highest success rate, while AHA-WAM remains second in success and achieves the highest progress score (35.00). This suggests that, despite not relying on $\pi_{0.5}$-scale generalist pretraining, AHA-WAM attains comparable real-world robustness and delivers the strongest deployment performance among WAM-based baselines. These results indicate that asynchronous world-action modeling improves deployment stability: long-horizon planner context supports task progress, while high-frequency execution enables timely physical correction.

4.5 Latency and Control Frequency

Finally, we compare closed-loop inference latency in Tab. 3. Fast-WAM is reported with official latency, while the remainings are measured on a single NVIDIA RTX 5090D GPU. Appendix D gives protocol and optimization details.

Table 3: Inference latency and frequency.

Method	Lat. (ms)	Freq. (Hz)	Speedup
Motus	1866.10	0.54	$0.10\times$
Fast-WAM	190.00	5.26	$1.00\times$
AHA-WAM	41.37	24.17	$4.59\times$
AHA-WAM-Flash	**17.56**	**56.95**	$10.82\times$

Compared with Fast-WAM, AHA-WAM reduces latency from 190.00 ms to 41.37 ms, improving the closed-loop rate from 5.26 Hz to 24.17 Hz. With the distilled sampler, AHA-WAM-Flash further reaches 17.56 ms and 56.95 Hz, yielding a $10.82\times$ speedup while keeping the same asynchronous planner–executor interface.

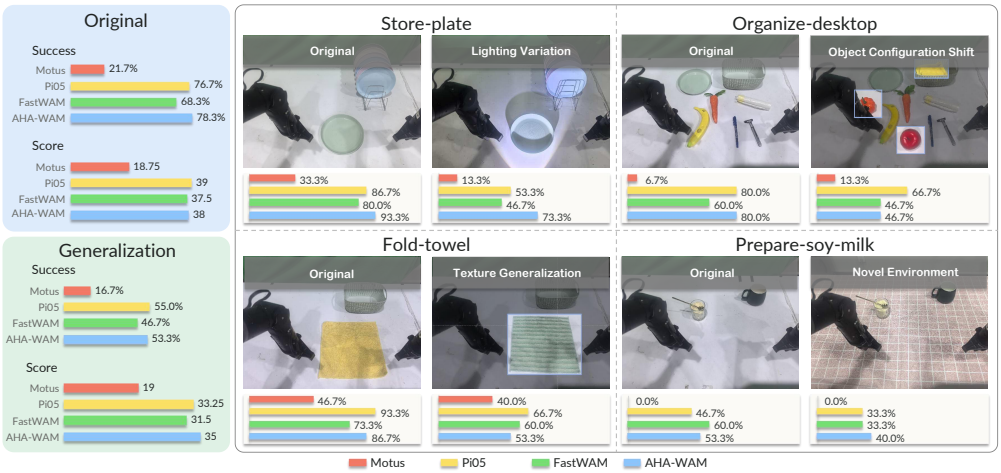

Figure 4: **Real-world task success rates and scores.** Success and 0–3 task scores are computed over 30 trials; scoring criteria are in Appendix C.

5 Conclusion

We presented AHA-WAM, an asynchronous horizon-adaptive world-action model that decouples a low-frequency video DiT world planner from a high-frequency action DiT closed-loop executor. Observation-Guided Video-Context Routing and horizon-adaptive offset training make this asynchronous interface effective by adapting reused planner context to each current observation and handling arbitrary planner-executor phase offsets. Experiments on RoboTwin and real-world manipulation tasks demonstrate strong performance and substantially reduced closed-loop latency, showing that learned visual dynamics can improve robot control without compromising control frequency.

Limitations and future work. The planner update frequency, video horizon, and action chunk size introduce temporal hyperparameters whose optimal allocation may depend on task dynamics and embodiment. Future work could strengthen the slow planner with longer-horizon prediction and richer scene representations, while systematic evaluation on dedicated long-horizon benchmarks would better quantify how the asynchronous design scales with task complexity.

More broadly, AHA-WAM opens a new design space for WAMs. Once the video branch is decoupled from the high-frequency control loop, it can afford more expensive computation without directly increasing action latency. The asynchronous interface makes these extensions particularly attractive: the video branch can become more predictive, deliberative, or physically grounded, while the action branch preserves the fast closed-loop rate needed for deployment. We believe this separation between slow scalable world planning and fast reactive action execution is a promising direction for building more capable, efficient, and deployable world-action models.

Acknowledgments

We would like to express our sincere gratitude to the Baidu AI Cloud Baige Team for their exceptional technical support and for providing access to the state-of-the-art Baidu AIHC platform. We specifically appreciate the platform's powerful capabilities in delivering efficient training acceleration and enabling ultra-low-latency inference, which were instrumental in optimizing our system's performance during evaluation. The advanced system optimizations and robust distributed infrastructure provided by this team were crucial in accelerating our experiments and validating the scalability of our proposed methods.

References

[1] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *CoRL*, 2023.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] T. Yang, G. Chen, Y. Chen, Z. Liang, Y. Liu, Z. Chen, C. Xu, H. Liang, J. Pang, Y. Mu, et al. HiVLA: A visual-grounded-centric hierarchical embodied manipulation system. *arXiv preprint arXiv:2604.14125*, 2026.

[4] Z. Liang, Y. Mu, H. Ma, M. Tomizuka, M. Ding, and P. Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *CVPR*, 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *RSS*, 2023.

[6] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

[7] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2025.

[8] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[9] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[10] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *ICLR*, 2025.

[11] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[12] Q. Bu, Y. Yang, J. Cai, S. Gao, G. Ren, M. Yao, P. Luo, and H. Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[13] Z. Liang, Y. Li, T. Yang, C. Wu, S. Mao, T. Nian, L. Pei, S. Zhou, X. Yang, J. Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. In *ICML*, 2026.

[14] J. Liu, H. Chen, P. An, Z. Liu, R. Zhang, C. Gu, X. Li, Z. Guo, S. Chen, M. Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025.

[15] J. Chen, W. Song, P. Ding, Z. Zhou, H. Zhao, F. Tang, D. Wang, and H. Li. Unified diffusion vla: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025.

[16] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *IJRR*, 2025.

[17] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

[18] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *ICML*, 2023.

[19] S. Gao, W. Liang, K. Zheng, A. Malik, S. Ye, S. Yu, W.-C. Tseng, Y. Dong, K. Mo, C.-H. Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

[20] D. Hafner et al. DreamerV3: Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[21] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[22] T. Seedance, D. Chen, L. Chen, X. Chen, Y. Chen, Z. Chen, Z. Chen, F. Cheng, T. Cheng, Y. Cheng, et al. Seedance 2.0: Advancing video generation for world complexity. *arXiv preprint arXiv:2604.14148*, 2026.

[23] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *NeurIPS*, 2023.

[24] Y. Tian, S. Yang, J. Zeng, P. Wang, D. Lin, H. Dong, and J. Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *ICLR*, 2025.

[25] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *ICML*, 2025.

[26] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[27] H. Bi, H. Tan, S. Xie, Z. Wang, S. Huang, H. Liu, R. Zhao, Y. Feng, C. Xiang, Y. Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[28] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[29] L. Li, Q. Zhang, Y. Luo, S. Yang, R. Wang, F. Han, M. Yu, Z. Gao, N. Xue, X. Zhu, Y. Shen, and Y. Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

[30] T. Yuan, Z. Dong, Y. Liu, and H. Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

[31] A. Ye, B. Wang, C. Ni, G. Huang, G. Zhao, H. Li, H. Li, J. Li, J. Lv, J. Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026.

[32] Q. Bu, H. Li, L. Chen, J. Cai, J. Zeng, H. Cui, M. Yao, and Y. Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

[33] H. Xue, J. Ren, W. Chen, G. Zhang, F. Yuan, G. Gu, H. Xu, and C. Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *ICRA 2025 Workshop*, 2025.

[34] N. Hirose, C. Glossop, D. Shah, and S. Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge. *arXiv preprint arXiv:2602.13476*, 2026.

[35] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li, Q. Liang, X. Lin, Y. Ge, Z. 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.

[36] Y. Yang, S. Zeng, T. Lin, X. Chang, D. Qi, J. Xiao, H. Liu, R. Chen, Y. Chen, D. Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

[37] S. Wu, X. Liu, S. Xie, P. Wang, X. Li, B. Yang, Z. Li, K. Zhu, H. Wu, Y. Liu, et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

[38] K. Black, M. Galliker, and S. Levine. Real-time execution of action chunking flow policies. *NeurIPS*, 2026.

A Implementation Settings

This appendix records implementation details that complement the experimental setup in Section 4.1. The main text describes the benchmark-level settings and the asynchronous planner–executor schedule; here we summarize the concrete implementation settings used for the RoboTwin runs. We omit logging paths, checkpoint paths, and inactive legacy annealing options.

Model architecture. Table 4 provides the key architecture and hyperparameter settings for AHA-WAM on RoboTwin, including the Video-DiT planner, Action-DiT executor, and OVCR interface. The instantiated model contains approximately 4.99B parameters in the video branch, 1.02B parameters in the action DiT, and 1.22B parameters in the memory and context-routing modules. During asynchronous inference, the action path receives the latest observation and proprioceptive state while querying the most recently available planner context through OVCR.

Table 4: Implementation settings for AHA-WAM on RoboTwin.

Component	Subcomponent	Setting
Component settings		
Video-DiT	Backbone	Wan2.2-TI2V-5B video expert
	Observation cameras	3 views: head, left wrist, right wrist
	Image resolution	384×320
	History frames	6
	Video/action frequency ratio	8
	Video RoPE stride	8
	Routed transformer layers	30
	Attention heads / head dimension	24 / 128
Action-DiT	Action horizon / chunk size	64 actions / 16 actions
	Number of action chunks	4
	State / proprio dimension	14
	Action dimension	14
	Training objective	Action prediction with planner-context conditioning
OVCR	Method name	Observation-Guided Video-Context Routing
	Query source	Visual observation context
	Queries per chunk	32
	Target context	Causal video K/V cache
	Granularity	Per action chunk, per transformer layer
Hyperparameter settings		
Training	Optimizer / learning rate / weight decay	AdamW / 1×10^{-4} / 1×10^{-2}
	Learning-rate schedule / warmup	Cosine / first 5% of training
	Global batch size / epochs	512 / 5
	Dataloader workers	16
	Video/action train timesteps	1000
	Video/action train and inference shift	5.0
Inference	Default action denoising steps / CFG	10 / 1.0

ODE distillation. Table 5 lists the hyperparameter settings used for ODE-distilled fast sampling. It records the distillation target, teacher–student denoising schedules, and the main optimization settings; dataset paths, checkpoint paths, logging paths, and other run-specific bookkeeping are omitted.

Table 5: Hyperparameter settings for ODE-distilled fast sampling.

Item	Setting
Distillation target	Action ODE / flow trajectory
Teacher denoising steps	16
Student denoising steps	2
Distilled timesteps	5,000
Teacher capture schedule	$0, 1, 2, 4, 8, 12, 16$
Prediction parameterization	Flow
Global batch size	512
Learning rate	2×10^{-5}
Optimizer / weight decay	AdamW / 1×10^{-2}
Learning-rate schedule	Cosine
Epochs	5
Local batch size / gradient accumulation	16 / 4
Dataloader workers	16

Training configuration. The optimizer, scheduler, batch-size, epoch, and default inference settings for AHA-WAM are summarized in Table 4, while the corresponding ODE-distillation settings are summarized in Table 5. We use flow matching for both video prediction and action prediction, with noise times sampled from a logit-normal distribution; when both branches are optimized jointly, the video loss and action loss are weighted equally. Horizon-adaptive offset training follows Section 3.3, so each action chunk is trained to consume planner context under randomized planner–executor phase offsets. The distilled AHA-WAM-Flash variant uses the reduced-step sampler described in Appendix D.

B RoboTwin Evaluation Details

This appendix provides the benchmark and baseline details omitted from the main RoboTwin discussion in Section 4.2.

Benchmark protocol. RoboTwin 2.0 is evaluated with the AgileX embodiment and contains 50 dual-arm manipulation tasks spanning a broad range of bimanual skills. Following the multi-task setting used by prior WAM baselines, AHA-WAM is trained with 50 clean demonstrations and 500 randomized demonstrations per task, totaling $2{,}500$ clean and $25{,}000$ randomized demonstrations. Each task is evaluated with 100 clean episodes and 100 randomized episodes, and we report task-averaged success rates. The randomized setting introduces visual and scene-level variations, providing a stronger robustness test than clean evaluation.

Baseline comparison. We compare AHA-WAM with representative VLA and WAM baselines. Fast-WAM is the closest baseline because it also uses a video DiT inside a world-action model, while Motus and LingBot-VA represent unified world-action modeling architectures. π_0, $\pi_{0.5}$, and ABot-M0 provide comparisons to strong generalist VLA policies. When available, we include both embodied-pretrained and Wan2.2-initialized variants to separate the effect of robot-data pretraining from the model architecture. In Table 1, Embodied PT. indicates whether the method uses embodied robot-data pretraining before RoboTwin 2.0 training. We report official published RoboTwin 2.0 results for external baselines when available.

C Real-World Task Execution and Scoring Criteria

Figure 5 provides a step-by-step illustration of the four real-world manipulation tasks used in our evaluation. Each task is decomposed into three ordered subtask steps, which are executed sequentially during a rollout and directly define the partial-progress score used in Figure 4. For each model and task, success rate and score are computed over 30 independent trials. Success is recorded as a binary outcome, while the score is assigned on a 0–3 scale: a score of 0 indicates no meaningful task progress, and scores of 1, 2, and 3 indicate completion of subtask steps respectively. The task is counted as success when the rollout reaches score 3. This unified execution-and-scoring protocol makes the evaluation more informative than a binary success label alone, because it distinguishes early failures from trials that complete most of the required manipulation sequence.

Together, these four tasks cover complementary real-world control challenges: rigid-object placement, deformable manipulation, long-horizon multi-object organization, and fine-grained tool use. Figure 5 and Table 6 therefore illustrate the task execution process and the basis for the 0–3 scoring criteria.

D Inference Speedup Details

Measurement protocol. We report L_{chunk}, the end-to-end latency of a single action-chunk inference call. This includes image encoding, incremental planner-context access, OVCR context routing, and Action DiT denoising. Video-DiT prefill runs asynchronously with action inference; we therefore use L_{chunk} as the primary latency metric for closed-loop action frequency and separately

Figure 5: Comprehensive illustration of the real-world task execution process. Each row corresponds to one task type, and the numbered panels show the ordered subtask steps executed in sequence.

Table 6: Summary of real-world task objectives, partial-progress scoring criteria, and evaluated control challenges. All tasks use a unified 0–3 score, where score 0 indicates no meaningful progress and score 3 indicates full task completion.

Task	Objective	Score	Completion criterion	Evaluated challenges
Store Plate	Move a plate into a plate rack.	1	The left hand securely grasps the plate.	Bimanual transfer, spatial placement, and pose control.
		2	The plate is handed over from the left hand to the right hand.	
		3	The right hand places the plate into the rack.	
Fold Towel	Fold a towel and place it into a basket.	1	Both hands fold the towel forward.	Deformable-object handling and robustness to shape changes.
		2	The right hand folds the towel once more.	
		3	The right hand deposits the folded towel into the basket.	
Organize Desktop	Clear multiple desktop objects into their target containers.	1	The left hand places left-side objects, such as the banana and carrot, onto the plate.	Multi-object sequencing, arm-specific selection, and subgoal tracking.
		2	The right hand places the pen and tool into the basket.	
		3	The right hand places the plastic bag roll into the basket.	
Prepare Soy Milk	Prepare soy milk powder using a cup and spoon.	1	The right hand moves the cup to the workspace center.	Fine-grained tool use, contact-rich manipulation, and bimanual coordination.
		2	The left hand scoops soy milk powder with the spoon and pours it into the cup.	
		3	The left hand stirs the mixture and returns the spoon.	

track prefill throughput to ensure that planner context can be refreshed in the background. All latency values are measured in milliseconds on RoboTwin 2.0 tasks unless otherwise specified. The first warmup episode is discarded, and remaining per-episode means are averaged. Both baseline and optimized pipelines use bf16 precision, so the reported speedups do not come from reducing numerical precision.

Table 7: Latency notation used in the inference speedup analysis.

Symbol	Definition
L_{chunk}	End-to-end latency of one action-chunk inference
L_{prefill}	Latency of one asynchronous Video-DiT prefill call

CUDA acceleration. Table 8 reports the cumulative optimization path from PyTorch eager inference to the optimized CUDA deployment. The main deployment optimizations are grouped into three categories. First, graph-level static capture compiles backbone modules including the Action DiT, memory/context modules, and VAE encoder into TensorRT engines and replays the fixed denoising loop with CUDA Graphs, reducing Python dispatch and redundant kernel launch overhead. Second, selective `torch.compile` is applied to the Video-DiT prefill path, where full trace export is difficult due to Python-side control flow. Third, hot-path redundancy elimination hoists chunk-level computations outside the denoising loop and skips repeated host-to-device copies for tensors that remain fixed across denoising steps.

Table 8: Cumulative CUDA acceleration ablation on RTX 5090D. CG denotes CUDA Graph.

Stage	Optimization	L_{chunk} (ms)	L_{prefill} (ms)	Runtime
0	PyTorch eager baseline	415.77 ± 0.33	61.15 ± 0.26	A
1	+ Action DiT TRT + CG	83.87 ± 0.51	–	A
2	+ Memory/context TRT + CG	71.37 ± 0.15	–	A
3	+ Video-DiT prefill `compile`	71.45 ± 0.14	34.59 ± 0.08	A
4	+ Hot-path redundancy elimination	50.37 ± 0.27	–	A
5	+ Runtime upgrade (A $\rightarrow$ B)	47.72 ± 0.08	–	B
6	+ TRT video-KV skip-copy	45.77 ± 0.03	–	B
7	+ VAE encoder TRT	41.37 ± 0.03	–	B

Runtime A: PyTorch 2.7.1 + cu128 / Triton 3.3.1 / TensorRT 10.16.1.11. Runtime B: PyTorch 2.12.0 + cu130 / Triton 3.7.0 / TensorRT 10.16.1.11.

Prefill compilation. The goal of prefill optimization is to make planner-context refresh available fast enough for asynchronous deployment. Table 9 compares compile modes. Although `reduce-overhead` gives the lowest prefill latency, it increases L_{chunk}; we therefore use the default mode.

Table 9: Video-DiT prefill compile mode comparison.

Configuration	L_{chunk} (ms)	L_{prefill} (ms)
Eager (no compile)	71.37	57.15 ± 0.14
compile (default)	71.45	34.59 ± 0.08
compile (reduce-overhead)	75.08	25.70 ± 0.22

Hot-path streamlining. The denoising loop contains computations that depend only on chunk-level inputs, such as condition embeddings, positional encodings, and references to planner-context K/V tensors. Hoisting these computations outside the 10-step loop reduces L_{chunk} from 71.45 ms to 63.25 ms. Removing redundant recursive state traversals over modules already in evaluation mode further reduces latency to 50.37 ms. The TensorRT wrapper also reuses static input buffers for video-KV tensors that remain unchanged within a chunk, reducing latency by another 1.95 ms in the 10-step setting.

VAE encoder validation. We validate the VAE encoder TensorRT engine against a PyTorch eager reference using both random inputs and real RoboTwin frames. As shown in Table 10, the TensorRT output remains numerically close to the reference and produces no NaN or Inf values.

ODE distillation. After CUDA acceleration establishes the optimized 10-step path, ODE distillation reduces the number of action denoising steps needed by the sampler. For AHA-WAM-Flash, we freeze the video DiT and distill only the action denoising path so that the student preserves the same planner-context and OVCR interface as the 10-step teacher. The teacher produces a 16-step denoising trajectory, indexed from the noisy state 0 to the final denoised state 16, and we select trajectory

Table 10: VAE encoder numerical validation against PyTorch eager reference.

Metric	Random inputs ($N = 1000$)	Real frames ($N = 1000$)
Cosine similarity (mean)	0.999974	0.999941
Cosine similarity (min)	0.999897	0.999589
Max absolute deviation (min–max)	0.031–0.078	0.023–0.086
Relative max deviation (mean)	1.24%	1.31%
NaN/Inf count	0	0

anchors $\{0, 1, 2, 4, 8, 12, 16\}$. During training, the student randomly samples a non-final anchor as the starting state and directly predicts the teacher's final denoised action state using a regression loss. We sample more frequently from states near the noisy end of the trajectory, because accurate prediction from high-noise states is the key requirement for reducing the number of inference denoising steps.

Table 11: Latency of the CUDA-accelerated action sampler under different ODE-distilled denoising step counts.

Denoising steps	L_{chunk} (ms)	Frequency (Hz)	Latency vs. 10-step
1	14.67	68.3	-64.60%
2	17.56	56.9	-57.50%
4	23.45	42.6	-43.30%
10	41.37	24.2	–

E Per-Task RoboTwin Success Rates

Table 12 reports per-task success rates on RoboTwin 2.0 for AHA-WAM and the selected baselines. All values are percentages, with clean and randomized evaluation reported separately.

Table 12: Per-task success rates on RoboTwin 2.0 under clean and randomized evaluation settings.

Task	AHA-WAM		AHA-WAM-Flash		Fast-WAM		LingBot-VA		$\pi_{0.5}$		Motus	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust Bottle	100%	100%	100%	100%	100%	100%	90%	94%	100%	99%	89%	93%
Beat Block Hammer	100%	94%	100%	93%	99%	97%	96%	98%	96%	93%	95%	88%
Blocks Ranking RGB	100%	98%	100%	100%	100%	100%	99%	98%	92%	85%	99%	97%
Blocks Ranking Size	98%	98%	97%	91%	94%	98%	94%	96%	49%	26%	75%	63%
Click Alarmclock	100%	100%	100%	100%	100%	100%	99%	100%	98%	89%	100%	100%
Click Bell	100%	100%	100%	100%	100%	100%	100%	100%	99%	66%	100%	100%
Dump Bin Bigbin	99%	92%	95%	95%	97%	96%	89%	96%	92%	97%	95%	91%
Grab Roller	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Handover Block	96%	85%	97%	87%	95%	81%	99%	78%	66%	57%	86%	73%
Handover Mic	99%	99%	94%	92%	99%	100%	94%	96%	98%	97%	78%	63%
Hanging Mug	85%	78%	67%	53%	58%	62%	40%	28%	18%	17%	38%	38%
Lift Pot	100%	100%	100%	100%	100%	100%	100%	99%	96%	85%	96%	99%
Move Can Pot	78%	83%	89%	94%	90%	88%	94%	97%	51%	55%	34%	74%
Move Pillbottle Pad	100%	99%	96%	98%	100%	99%	99%	99%	84%	61%	93%	96%
Move Playingcard Away	100%	100%	98%	100%	100%	100%	100%	99%	96%	84%	100%	96%
Move Stapler Pad	88%	77%	75%	63%	77%	64%	91%	79%	56%	42%	83%	85%
Open Laptop	98%	97%	98%	98%	98%	100%	92%	94%	90%	96%	95%	91%
Open Microwave	40%	46%	9%	23%	62%	45%	82%	86%	34%	77%	95%	91%
Pick Diverse Bottles	89%	85%	83%	87%	80%	85%	89%	82%	81%	71%	90%	91%
Pick Dual Bottles	98%	99%	96%	98%	100%	96%	100%	99%	93%	63%	96%	90%
Place A2B Left	100%	92%	96%	98%	95%	93%	97%	93%	87%	82%	88%	79%
Place A2B Right	96%	92%	98%	96%	93%	99%	97%	95%	87%	84%	91%	87%
Place Bread Basket	89%	94%	89%	98%	91%	93%	97%	95%	77%	64%	91%	94%
Place Bread Skillet	91%	87%	87%	85%	90%	93%	95%	90%	85%	66%	86%	83%
Place Burger Fries	99%	98%	94%	98%	96%	99%	97%	95%	94%	87%	98%	98%
Place Can Basket	81%	94%	83%	79%	71%	69%	81%	84%	62%	62%	81%	76%
Place Cans Plasticbox	100%	100%	98%	100%	99%	96%	100%	99%	94%	84%	98%	94%
Place Container Plate	100%	96%	100%	94%	96%	100%	99%	97%	99%	95%	98%	99%
Place Dual Shoes	91%	99%	92%	89%	94%	88%	94%	89%	75%	75%	93%	87%
Place Empty Cup	100%	100%	100%	98%	100%	100%	100%	100%	100%	99%	99%	98%
Place Fan	100%	87%	96%	94%	96%	96%	99%	93%	87%	85%	91%	87%
Place Mouse Pad	93%	82%	79%	81%	83%	89%	93%	96%	60%	39%	66%	68%
Place Object Basket	81%	80%	89%	85%	89%	88%	91%	88%	80%	76%	81%	87%
Place Object Scale	96%	87%	89%	77%	90%	97%	96%	95%	86%	80%	88%	85%
Place Object Stand	94%	93%	96%	90%	90%	94%	99%	96%	91%	85%	98%	97%
Place Phone Stand	98%	96%	98%	94%	97%	99%	97%	97%	81%	81%	87%	86%
Place Shoe	91%	96%	94%	98%	96%	99%	98%	98%	92%	93%	99%	97%
Press Stapler	98%	99%	98%	100%	90%	97%	85%	82%	87%	83%	93%	98%
Put Bottles Dustbin	83%	96%	83%	77%	95%	90%	87%	91%	84%	79%	81%	79%
Put Object Cabinet	91%	89%	89%	85%	94%	89%	85%	87%	80%	79%	88%	71%
Rotate QRcode	90%	90%	88%	90%	93%	89%	96%	91%	89%	87%	89%	73%
Scan Object	94%	90%	85%	91%	89%	92%	96%	91%	72%	65%	67%	66%
Shake Bottle	100%	100%	100%	100%	100%	100%	100%	97%	99%	97%	100%	97%
Shake Bottle Horizontally	100%	100%	100%	100%	100%	100%	100%	99%	99%	99%	100%	98%
Stack Blocks Three	99%	98%	100%	98%	95%	97%	99%	98%	91%	76%	91%	95%
Stack Blocks Two	100%	99%	100%	100%	100%	100%	100%	98%	97%	100%	100%	98%
Stack Bowls Three	79%	82%	81%	83%	80%	81%	86%	83%	77%	71%	79%	87%
Stack Bowls Two	100%	100%	94%	98%	92%	98%	94%	98%	95%	96%	98%	98%
Stamp Seal	98%	90%	81%	83%	90%	94%	96%	97%	79%	55%	93%	92%
Turn Switch	70%	74%	53%	65%	61%	59%	44%	45%	62%	54%	84%	78%
Average	93.40%	92.20%	90.48%	89.92%	91.88%	91.78%	92.90%	91.50%	82.74%	76.76%	88.66%	87.02%