RAPID-WAM: RETHINKING THE WORLD–ACTION INTERFACE FOR COMPACT ACTION DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) use the rich representations of video-pretrained world models for robot control, but existing methods typically rely on large action experts and dense world–action interactions. Through feature interventions across three representative WAM formulations, we find that selectively retaining approximately one quarter of the most action-sensitive world features preserves closed-loop performance close to full access. Based on this observation, we propose RAPID-WAM with a lightweight world–action interface that uses task-conditioned queries to read and compress representations from multiple world-model depths into compact control conditions before action decoding. RAPID-WAM outperforms the corresponding Fast-WAM baselines across action-only, inverse-dynamics, and joint world–action formulations and achieves state-of-the-art performance on multiple robot manipulation benchmarks. In the default action-only setting, RAPID-WAM reduces the world-condition representation delivered to action decoding by $\mathbf{11{,}760}\times$, improves LIBERO-Plus success from **49.7%** to **70.7%**, and achieves a $\mathbf{4.53}\times$ speedup in end-to-end policy inference.

1 INTRODUCTION

Vision–language–action models (VLAs) build on large-scale vision–language pretraining and diverse robot demonstrations to predict actions from current observations and language instructions, and have become an important paradigm for general-purpose robot control (Brohan et al., 2023; Black et al., 2024; Physical Intelligence et al., 2025). World action models (WAMs) offer a complementary direction by leveraging video-pretrained world models whose representations capture object motion, interactions, and scene dynamics (Ye et al., 2026; Li et al., 2026; Kim et al., 2026). These representations allow robot control to draw on information about environment dynamics that is difficult to acquire directly from static visual supervision.

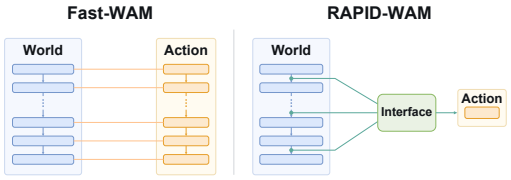

Figure 1: World–action interfaces.

Existing world action models (WAMs) connect world modeling and action prediction through several representative formulations, including action-only models such as Fast-WAM (Yuan et al., 2026), inverse-dynamics models such as LingBot-VA (Li et al., 2026), and joint world–action models such as DreamZero and Motus (Ye et al., 2026; Bi et al., 2025). Despite these differences, existing WAMs typically rely on large action experts and dense world–action interactions to use rich world representations. Through feature interventions across three representative WAM formulations, we find that action prediction depends highly unevenly on world representations: selectively retaining approximately one quarter of the most sensitive features preserves closed-loop performance close to full access (Figure 2). This observation raises the question of how world representations should be delivered to action prediction.

To address this question, we propose **RAPID-WAM** (**R**ethinking **A**ction **P**rediction via **I**nterface **D**esign in World Action Models), whose core design is a lightweight world–action interface that selectively reads world representations before action decoding (Figure 1). The interface uses task-

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.

conditioned queries to read and compress representations from multiple world-model depths into compact control conditions before action decoding. We instantiate this design in action-only, inverse-dynamics, and joint world–action formulations. Across multiple simulation benchmarks, RAPID-WAM outperforms the corresponding Fast-WAM baselines in all three formulations, further improves performance on real-world tasks, and substantially reduces action-side computation under unified profiling. Under the default action-only setting, RAPID-WAM reduces the world-condition representation delivered to the action decoder by $\mathbf{11{,}760}\times$, improves LIBERO-Plus success from **49.7%** to **70.7%**, and achieves a $\mathbf{4.53}\times$ speedup in end-to-end inference.

Our main contributions are as follows:

- We revisit WAMs from the perspective of the world–action interface. Controlled interventions across three representative WAM formulations reveal that action prediction depends highly unevenly on world representations.
- We propose **RAPID-WAM**, whose core design is a lightweight world–action interface that uses task-conditioned queries to read and compress multi-depth world representations into compact control conditions before action decoding.
- We demonstrate consistent control gains across three representative WAM formulations in simulation, further improvements on real-world tasks, and substantial reductions in action-side computation under unified profiling.

2 Related Work

World Action Models. World action models (WAMs) adapt video-pretrained generative models for robot control, allowing action learning to use representations formed through modeling visual dynamics (Ye et al., 2026; Li et al., 2026; Kim et al., 2026). One key difference among existing methods is how future world states contribute to action prediction. LingBot-VA uses causal video–action modeling, allowing predicted visual dynamics to support subsequent action generation (Li et al., 2026). DreamZero jointly models future video and robot actions, while Motus uses a Mixture-of-Transformers architecture to support multiple forms of video–action modeling within a unified model (Ye et al., 2026; Bi et al., 2025). Cosmos Policy encodes actions together with future states and values as latent frames within a pretrained video model and further supports model-based planning (Kim et al., 2026). Fast-WAM retains video co-training but removes explicit future generation at inference, directly using world representations of the current observation to predict actions (Yuan et al., 2026). These works explore different ways to incorporate world modeling into robot control. Our work addresses the complementary question of how representations produced by the world model should be delivered to action prediction.

World–Action Coupling. Existing WAMs use different mechanisms to couple world representations with robot actions. Mixture-of-Transformers models such as LingBot-VA, Motus, and Fast-WAM connect modality-specific computation streams through shared attention, allowing action representations to interact directly with visual world representations (Li et al., 2026; Bi et al., 2025; Yuan et al., 2026). DiT4DiT extracts intermediate representations from a video Diffusion Transformer and provides them to a separate action DiT through cross-attention, while showing that the choice of feature extraction depth substantially affects control performance (Ma et al., 2026). These methods demonstrate the value of representations learned through video modeling for action prediction. However, in designs with separate action networks, the action module typically still accesses spatial world representations directly through repeated attention, rather than first forming compact control representations. Meanwhile, lightweight diffusion policies such as HDP3 show that continuous actions can be decoded effectively with compact action networks (Zhang et al., 2026b). RAPID-WAM connects these two directions by studying how to selectively read and compress world representations before action decoding, thereby supporting lightweight action decoders across action-only, inverse-dynamics, and joint world–action formulations.

3 Diagnosing World–Action Access

A common design in existing WAMs is to expose rich world representations directly to the action predictor. Mixture-of-Transformers models such as LingBot-VA, Motus, and Fast-WAM couple the

world and action streams through shared or joint attention, while DiT4DiT allows its action DiT to directly read intermediate features from the video model through cross-attention (Li et al., 2026; Bi et al., 2025; Yuan et al., 2026; Ma et al., 2026). In these architectures, the action network must both generate control outputs and identify control-relevant information through repeated access to the available world representations. The selection of world information and action decoding therefore fall to the same deep action module, placing a substantial representation-processing burden on the action side. We therefore ask whether action prediction really requires such broad access to the full world representation.

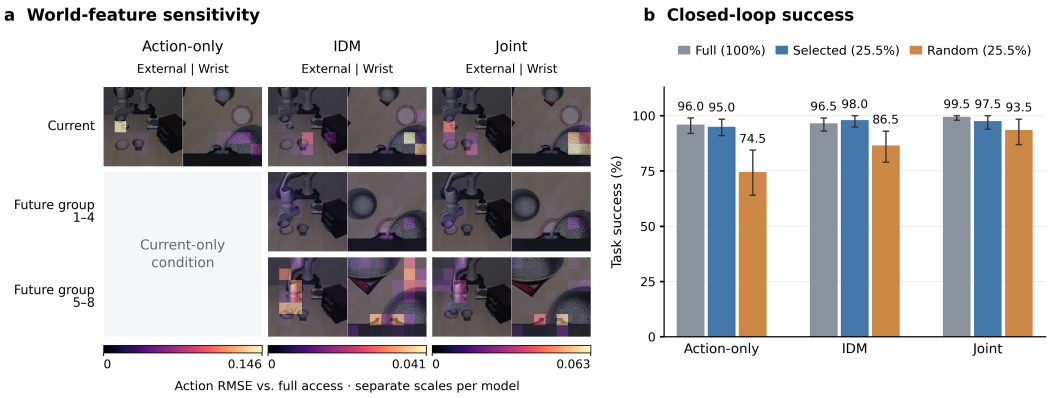

Figure 2: **Diagnosing world-feature access across three WAM formulations.** (a) Sensitivity of action predictions to interventions on individual world-feature positions. Brighter regions indicate larger changes. (b) Closed-loop success with full access and with 25.5% of positions retained by sensitivity-based or random selection.

To examine whether this behavior persists across different ways of incorporating world information, we analyze three trained WAMs covering action-only, inverse-dynamics, and joint world–action formulations in the Fast-WAM family. We first visualize how strongly the final action prediction depends on each spatial world-feature position. For one position at a time, we replace its world keys and values with their spatial mean from the same layer and temporal group and apply the intervention throughout the world–action interactions. We then measure the resulting change in the final action prediction. Figure 2(a) reveals the same pattern across all three formulations. Action prediction depends highly unevenly on the available world features. Only part of the representation strongly affects the predicted action, while interventions on many other positions produce much smaller changes. This uneven dependence remains when future representations are introduced in IDM and when world and action states are generated jointly.

This observation suggests that the action network may not need access to the complete world representation. We therefore retain only 25.5% of the available world positions according to their measured sensitivity and remove action-side access to the remainder. Figure 2(b) shows that this selected quarter preserves performance close to full access in all three formulations. Success changes from 96.0% to 95.0% for the action-only model, from 96.5% to 98.0% for IDM, and from 99.5% to 97.5% for the joint model. Thus, approximately one quarter of the available world positions is sufficient to recover near-full closed-loop performance, with the IDM model even obtaining a higher point estimate after selection. We also find that sensitivity-based feature selection matters. When the same number of positions is retained at random, success drops to 74.5%, 86.5%, and 93.5% in the action-only, IDM, and joint models, respectively. Full intervention protocols and diagnostic results are provided in Appendix A.

These results suggest that access to the full world representation is substantially redundant for action prediction. By actively selecting features more relevant to control, we can preserve closed-loop performance close to full access using only about one quarter of the available world features. Motivated by this observation, we argue that a dedicated lightweight interface can actively read control-relevant representations from the world model and compress them into compact action conditions before passing them to the action network for control prediction. The action network would then no longer need to repeatedly search the full world representation for useful information. Instead, it could focus

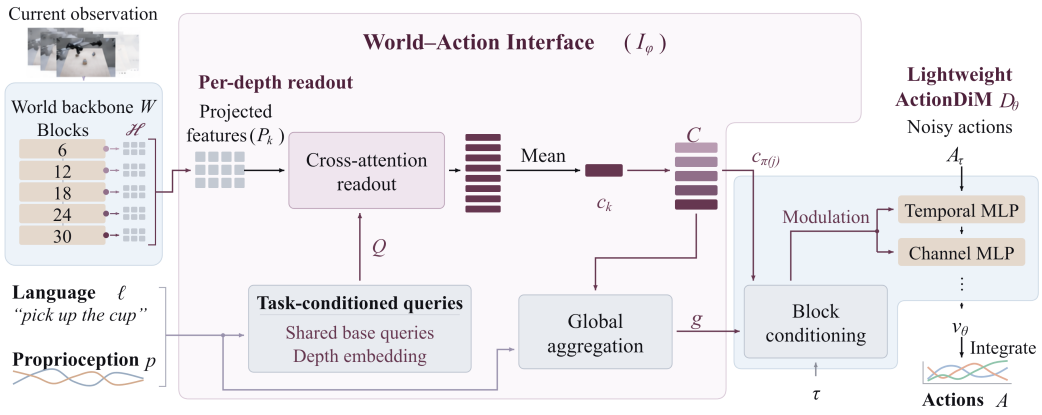

Figure 3: **Overview of RAPID-WAM.** Task-conditioned queries read and compress multi-depth world representations into compact conditions that modulate a lightweight action decoder.

on transforming the resulting control conditions into action outputs, allowing the action model itself to be substantially smaller. Based on this idea, we propose RAPID-WAM, whose detailed design is presented in the next section.

4 RAPID-WAM

4.1 Overview

RAPID-WAM places a lightweight learned interface between the world backbone and the action decoder. The interface uses task-conditioned queries to read representations from multiple world-model depths and compresses them into depth-specific conditions together with a global condition. The action decoder receives world information only through these compact conditions and does not directly attend to spatial world representations.

Let $\mathcal{H} = \{H_{\ell_k}\}_{k=1}^{K}$ denote the hidden states from the selected world-model depths. Given a language instruction ℓ and proprioceptive state p, the interface produces

$$(C, g) = I_\phi(\mathcal{H}, \ell, p),$$

where C contains the depth-specific conditions and g is the global condition. Given a noisy action chunk A_τ at noise level τ, the action decoder predicts the action flow

$$v_\theta = D_\theta(A_\tau, \tau; C, g).$$

Integrating this flow produces the final action chunk $A \in \mathbb{R}^{T \times d_{\text{act}}}$. Figure 3 illustrates the overall architecture.

4.2 World–Action Interface

We first describe the default action-only formulation of RAPID-WAM. Temporal extensions for IDM and joint models are detailed in Appendix B.3. In MoT-based WAMs such as Fast-WAM, the world and action streams interact through shared attention within Transformer blocks (Yuan et al., 2026). Instead, RAPID-WAM reads the outputs of a small set of selected world-model blocks. The world backbone first forms its representations, after which a lightweight interface extracts and compresses control-relevant information before it reaches the action decoder. This avoids coupling a deep action expert to the spatial world tokens throughout the world backbone.

Per-depth Readout. Let

$$H_{\ell_k} \in \mathbb{R}^{n_k \times d_w}$$

denote the output of the k-th selected world-model block. We normalize and project each representation to a shared condition width

$$P_k = \text{Proj}_k\left(\text{LN}_k(H_{\ell_k})\right).$$

Rather than passing the spatial features in P_k directly to the action decoder, the interface reads them with a small set of task-conditioned queries.

We map the backbone conditioning context and proprioceptive state to embeddings $e_\ell, e_p \in \mathbb{R}^{d_c}$ and associate each selected depth with a learned embedding $e_k \in \mathbb{R}^{d_c}$. For depth k, the i-th query is

$$q_{k,i} = q_i^{\text{base}} + f_q(e_\ell + e_p + e_k), \qquad i = 1, \dots, Q. \tag{1}$$

The shared base queries provide learned readout slots, while language, proprioception, and depth information adapt the readout to the current control problem.

After stacking the queries into Q_k, we use cross-attention to read the projected world representation

$$\begin{aligned} U_k &= \mathrm{MHA}(Q_k, P_k, P_k), \\ R_k &= \mathrm{LN}_{\text{out}}\left(U_k + f_r(U_k)\right), \\ c_k &= \frac{1}{Q}\sum_{i=1}^{Q} R_{k,i}. \end{aligned} \tag{2}$$

Each spatial representation is therefore compressed into a fixed-width condition $c_k \in \mathbb{R}^{d_c}$. The readout performs this compression before action decoding, so the action network never needs to search the full spatial representation itself. By default, we read five evenly spaced world blocks $\{6, 12, 18, 24, 30\}$ using $Q = 8$ queries per depth and $d_c = 256$.

Depth-specific and Global Conditions. The interface retains the summaries from the selected depths as

$$C = (c_1, \dots, c_K)$$

and constructs a shared global condition

$$g = f_g\left(\frac{1}{K}\sum_{k=1}^{K} c_k + e_\ell + e_p\right). \tag{3}$$

The depth-specific conditions preserve separate summaries from different stages of world-model computation, while the global condition aggregates information across depths together with language and proprioception. Together, C and g form the compact representation communicated across the world–action interface.

Lightweight Action Decoding. Because spatial world representations have already been read and compressed by the interface, action generation no longer requires dense attention over world tokens. We therefore use a lightweight ActionDiM based on the temporal and channel mixing architecture of HDP3 (Zhang et al., 2026b).

For decoder block j, a fixed depth schedule supplies one depth-specific condition $c_{\pi(j)}$. We combine it with the global condition and the action noise-time embedding

$$m_j = G_j\left([c_{\pi(j)}, g, e_\tau]\right), \tag{4}$$

where G_j produces the adaptive normalization parameters for the temporal and channel mixing branches (Peebles & Xie, 2023). World information thus enters the action network only through the compact conditions C and g, rather than through direct attention to spatial world tokens. Our default decoder contains eight blocks with width 256. The detailed block structure and depth assignment are provided in Appendix B.1.

4.3 Training and Inference

We jointly optimize the world backbone, interface, and action decoder. Following the flow-matching formulation used in recent WAMs, given an action chunk A, Gaussian noise ϵ, and noise level τ, we construct

$$A_\tau = (1 - \tau)A + \tau\epsilon$$

and train the action decoder to predict the corresponding flow

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{A,\epsilon,\tau}\left[w(\tau)\left\|v_\theta(A_\tau, \tau; C, g) - (\epsilon - A)\right\|^2\right]. \tag{5}$$

We retain the video objective of the world backbone and optimize

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{video}} \mathcal{L}_{\text{video}}.$$

Action gradients propagate through the interface into the world backbone, so the world representations and their readout are adapted jointly for control.

In the default action-only formulation, the world backbone processes the current observation and the interface constructs the compact conditions C and g once per replan. The action chunk is initialized from Gaussian noise and generated by integrating the predicted flow, with the conditions held fixed throughout action sampling. We use ten action sampling steps by default.

In the inverse-dynamics formulation, compact conditions are constructed from current and generated future representations after future generation and reused throughout action sampling. In the joint world–action formulation, the conditions are refreshed at each joint denoising step as the future representations evolve. All three formulations use the same spatial readout design and lightweight action-decoder architecture, with formulation-specific temporal conditioning. Optimization details are provided in Appendix C.1; sampling and formulation-specific procedures are detailed in Appendices B.2–B.4.

5 Experiments

We evaluate RAPID-WAM along three dimensions. First, we assess its control performance and robustness on simulation and real-robot tasks. Next, we compare action-side computation and inference latency across three representative WAM formulations. Finally, we analyze the main design choices in the compact world–action interface.

5.1 Experimental Setup

Base model and training. We build RAPID-WAM on the pretrained Wan2.2-TI2V-5B world model used by the Fast-WAM family (Yuan et al., 2026). Unless otherwise stated, all models trained in our simulation experiments are optimized for 21,700 steps with an effective batch size of 128. Full optimization settings and formulation-specific training details are provided in Appendices C.1 and B.4.

Benchmarks and evaluation. We evaluate RAPID-WAM on LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2025), RoboCasa-24 Human-50 (Nasiriany et al., 2024), and real-world manipulation tasks. On LIBERO, we combine demonstrations from the Spatial, Object, Goal, and Long suites to train a unified policy and evaluate all 40 tasks with 50 rollouts per task. We then evaluate the same LIBERO-trained checkpoint directly on all 10,030 LIBERO-Plus variants without any additional adaptation. On RoboCasa, we train separate policies using the Human-50 data and evaluate on split B. Each model is evaluated on 24 tasks with 50 rollouts per task, totaling 1,200 rollouts per model. We use success rate as the primary measure of control performance. Complete training and evaluation protocols and real-world experimental details are provided in Appendices C and F.

Baselines and comparisons. We compare with representative VLA and WAM methods, using the Fast-WAM family as our primary reference baselines. On LIBERO and LIBERO-Plus, results for Fast-WAM and its IDM and Joint variants come from published evaluations; the corresponding Fast-WAM results for RoboCasa, real-robot tasks, and efficiency experiments come from our own experiments. Other baseline results are taken from the original papers or official releases. Specific result sources and model versions are documented in Appendix C.4.

5.2 Main Results

LIBERO. RAPID-WAM achieves the highest average success rate of **98.8%** on LIBERO, improving over Fast-WAM's 97.6% by 1.2 percentage points and outperforming strong WAM methods such as LingBot-VA and DiT4DiT. Success on Long increases from 95.2% to 97.8% (Table 1). Under both the IDM and joint formulations, RAPID-WAM also outperforms the corresponding Fast-WAM

Table 1: Success rates (%) on LIBERO.

Method	Spatial	Object	Goal	Long	Avg.
VLA-based methods					
OpenVLA	84.7	88.4	79.2	53.7	76.5
WorldVLA	87.6	96.2	83.4	60.0	81.8
π_0-FAST	96.4	96.8	88.6	60.2	85.5
NORA	92.2	95.4	89.4	74.6	87.9
π_0	96.8	98.8	95.8	85.2	94.1
UniVLA	96.5	96.8	95.6	92.0	95.2
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
VLA-Adapter-Pro	**99.6**	**99.6**	**98.2**	**96.4**	**98.5**
WAM-based methods					
Motus	96.8	99.8	96.6	97.6	97.7
LingBot-VA	98.5	99.6	97.2	**98.5**	98.5
DiT4DiT	98.4	99.6	**98.6**	97.6	98.6
Fast-WAM	98.2	**100.0**	97.0	95.2	97.6
RAPID-WAM	99.0	99.8	98.4	97.8	**98.8**
Fast-WAM-IDM	98.8	97.8	97.8	97.6	98.0
RAPID-WAM-IDM	**99.6**	**100.0**	97.6	97.2	98.6
Fast-WAM-Joint	**99.6**	99.4	98.2	96.8	98.5
RAPID-WAM-Joint	98.8	99.8	98.0	97.8	98.6

Table 2: Success rates (%) on RoboCasa-24 Human-50. †Overall recomputed using task-count weights; see Appendix C.4.

Method	Pick & Place	Doors/ Drawers	Others	Overall
BC-Transformer	3.8	46.7	38.0	28.8
VLA-based methods				
$\pi_{0.5}$	36.0	46.5	39.5	40.1
π_0	14.0	53.1	58.5	42.4
GeoPredict	22.7	**75.1**	**62.4**	52.4
GaussianDream	**43.8**	66.3	54.4	53.8†
Being-H0.5	36.0	71.7	57.6	**53.9**
WAM-based methods				
GWM	14.8	54.3	49.8	39.3
Fast-WAM	10.50	75.33	47.80	42.25
RAPID-WAM	**35.50**	85.00	54.40	55.75
Fast-WAM-IDM	10.00	75.67	49.60	42.92
RAPID-WAM-IDM	35.25	84.33	55.60	**56.00**
Fast-WAM-Joint	9.75	79.00	47.40	42.75
RAPID-WAM-Joint	30.25	**86.00**	**56.00**	54.92

Table 3: Success rates (%) across seven LIBERO-Plus perturbation categories. Robot, Lang., and BG denote robot initial state, language instruction, and background texture. Dashes indicate unavailable results.

Method	Camera	Robot	Lang.	Light	BG	Noise	Layout	Overall
VLA-based methods								
OpenVLA	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
WorldVLA	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
NORA	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
UniVLA	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
π_0	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
π_0-FAST	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
OpenVLA-OFT	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
$\pi_{0.5}$	**75.4**	**77.5**	**85.6**	**96.9**	**94.6**	**89.7**	**85.7**	**85.7**
WAM-based methods								
LDA-1B	–	–	–	–	–	–	–	45.5
Robust-WAM	28.7	43.4	73.6	90.2	**62.6**	56.8	67.8	58.9
LingBot-VA	40.9	**83.0**	86.4	82.3	53.1	64.4	76.2	69.5
Fast-WAM	16.5	43.4	67.8	80.1	51.6	37.9	61.2	49.7
RAPID-WAM	64.2	47.3	90.4	90.4	44.9	78.1	77.2	70.7
Fast-WAM-IDM	46.2	69.1	**94.2**	92.8	56.4	61.8	**81.5**	71.4
RAPID-WAM-IDM	**66.2**	49.5	89.2	92.2	48.1	**79.3**	77.8	72.0
Fast-WAM-Joint	38.7	63.9	93.1	**94.7**	55.7	57.7	77.5	68.1
RAPID-WAM-Joint	**66.2**	50.2	89.9	92.7	50.8	77.3	77.7	**72.3**

baselines. These results show that the compact world–action interface maintains strong control performance across all three WAM formulations.

LIBERO-Plus. We further evaluate zero-shot robustness on LIBERO-Plus. All our models are trained only on LIBERO and evaluated directly on LIBERO-Plus without additional adaptation. The default RAPID-WAM achieves **70.7%** overall success, improving over Fast-WAM's 49.7% by 21.0 percentage points and outperforming Fast-WAM in six of the seven perturbation categories (Table 3). Meanwhile, RAPID-WAM-IDM improves overall success from 71.4% to 72.0%, and RAPID-WAM-Joint improves it from 68.1% to **72.3%**. These results show that the compact interface of RAPID-WAM supports strong out-of-distribution generalization across different formulations under substantial distribution shift.

RoboCasa. We also evaluate all three WAM formulations of RAPID-WAM on RoboCasa-24 Human-50 to assess performance in complex household manipulation environments. All three formulations consistently improve over their corresponding Fast-WAM baselines. Overall success increases from 42.3% to 55.8% for action-only, from 42.9% to **56.0%** for IDM, and from 42.8% to 54.9% for joint world–action (Table 2), corresponding to gains of 13.5, 13.1, and 12.2 percentage points, respectively. These consistent improvements across all three formulations further show that the compact world–action interface effectively supports a range of complex household manipulation tasks.

Real-World Experiments. We further evaluate the default action-only RAPID-WAM against Fast-WAM on a bimanual robot platform equipped with two AgileX Piper arms, using two independent single-arm manipulation tasks, cup placement and plush toy placement, performed by the left and right arms, respectively. We collect 150 expert demonstrations per task and train a separate policy for each task and method. Fast-WAM and RAPID-WAM use the same demonstration data, and each method is evaluated with 40 physical rollouts per task. RAPID-WAM achieves success rates of 55.0% and 40.0% on the two tasks, improving over Fast-WAM by 10.0 and 7.5 percentage points, respectively. Average success increases from 38.75% to **47.50%**, a gain of 8.75 percentage points (Table 4). These results show that the compact world–action interface of RAPID-WAM supports closed-loop manipulation on a real robot. Task definitions, platform configuration, and evaluation details are provided in Appendix F.

Table 4: Real-world task success rates (%).

Task	Fast-WAM	RAPID-WAM
Cup placement	45.0	**55.0**
Plush toy placement	32.5	**40.0**
Average	38.75	**47.50**

5.3 Efficiency across WAM Formulations

We further evaluate how the compact world–action interface affects representation size and action-side efficiency across the three WAM formulations under a unified profiling setup. In the default action-only setting, Fast-WAM exposes 18,063,360 scalar elements of world-condition representations to the action decoder, whereas RAPID-WAM delivers only 1,536, a $\mathbf{11{,}760}\times$ reduction. In the IDM and joint formulations, the delivered representation is similarly reduced from 54,190,080 to 41,216 elements. This compact interface enables RAPID-WAM to replace Fast-WAM's 1.02B-parameter action expert with a 9.66M-parameter decoder and reduces action-stage FLOPs by $268.6\times$, $118.4\times$, and $30.7\times$ across the three formulations, respectively (Table 5).

These reductions further translate into faster inference. Action-generation latency decreases by $8.73\times$, $8.70\times$, and $4.69\times$ across the three formulations. In the action-only setting, end-to-end policy latency drops from 384.61 ms to 84.81 ms, a $\mathbf{4.53}\times$ speedup. Together with the control results in Tables 1–3, these results show that RAPID-WAM substantially compresses the world representation delivered to action decoding and reduces action-side computation while maintaining or improving control performance across the three WAM formulations.

Table 5: Delivered world-condition size and efficiency across WAM formulations.

Formulation	Method	Delivered world cond. (elements)	Decoder Params	Interface GFLOPs	Action GFLOPs	Action-gen. latency (ms)	Policy latency (ms)
Action-only	Fast-WAM	18,063,360	1.02B	257.26	1055.33	323.63	384.61
	RAPID-WAM	**1,536**	**9.66M**	**1.07**	**3.93**	**37.07**	**84.81**
IDM	Fast-WAM-IDM	54,190,080	1.02B	280.38	1078.45	321.55	785.90
	RAPID-WAM-IDM	**41,216**	**9.66M**	**6.25**	**9.11**	**36.94**	**485.83**
Joint	Fast-WAM-Joint	54,190,080	1.02B	280.38	1078.45	323.79	672.27
	RAPID-WAM-Joint	**41,216**	**9.66M**	**32.27**	**35.13**	**69.11**	**434.12**

5.4 Interface Design Analysis

We further examine the key design choices of the compact world–action interface. Figure 4(a,b) shows that reading representations from multiple depths improves performance under distribution

shift, while densely accessing all 30 world-model layers provides little additional benefit. A clearer difference comes from the readout mechanism. With the same five representation depths, learned readout achieves 70.7% on LIBERO-Plus, substantially outperforming mean pooling at 61.8%. This suggests that compactness alone is not sufficient. How control-relevant information is selected and summarized also matters. Results across different query counts further show that strong performance can be obtained with a small readout capacity, and we use eight queries as the default. Full suite and perturbation breakdowns are provided in Appendix D.2.

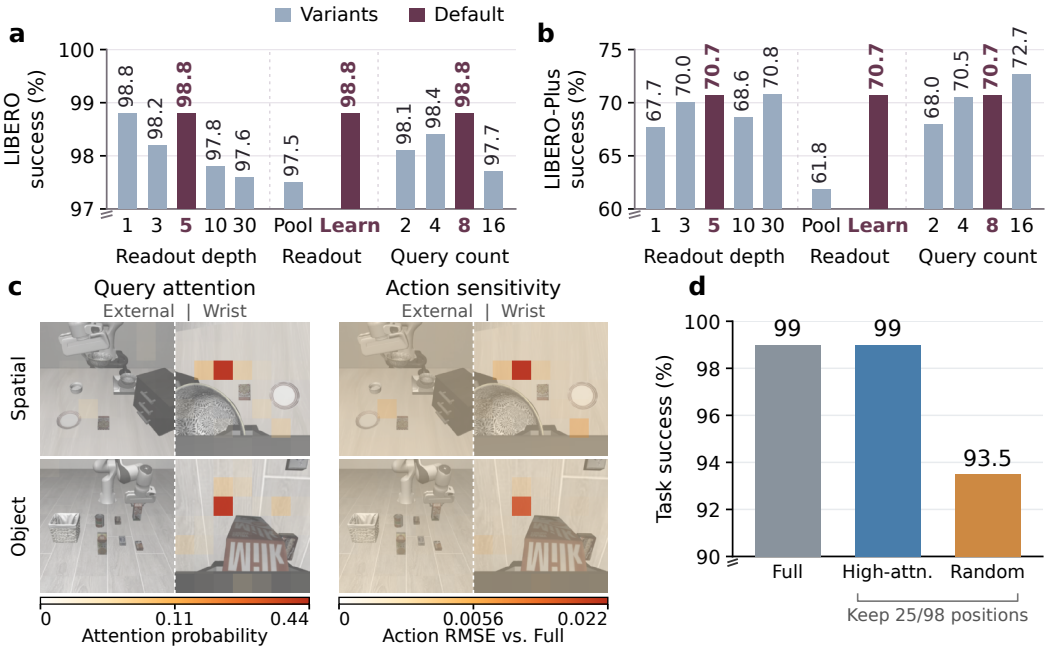

Figure 4: **Interface design and reader analysis.** (a,b) Readout ablations on LIBERO and LIBERO-Plus (2,000 and 10,030 rollouts per configuration). (c) Reader attention and action sensitivity. (d) Closed-loop retention diagnostic (200 paired episodes per condition).

Section 3 showed that Fast-WAM exposes the action network to many world features even though only a subset strongly affects action prediction. We therefore examine whether the learned reader in the default action-only model focuses on information useful for control. As shown in Figure 4(c), reader attention visually corresponds to regions with high action sensitivity under the intervention analysis. We then rank spatial positions by interface attention and retain only the top 25 of 98 positions during closed-loop control. As shown in Figure 4(d), this high-attention subset achieves 99.0% overall success, identical to full access, whereas retaining the same number of randomly selected positions yields only 93.5%. These results indicate that the learned reader prioritizes world features useful for closed-loop control.

6 Conclusion

We revisit how world representations are delivered to action prediction in WAMs. Through feature interventions across three representative WAM formulations, we find that action prediction depends highly unevenly on world representations, motivating a more selective world–action interface. We propose RAPID-WAM, whose core design uses task-conditioned queries to read and compress multi-depth world representations into compact control conditions before action decoding. Across simulation benchmarks, RAPID-WAM consistently improves over the corresponding Fast-WAM baselines in action-only, inverse-dynamics, and joint world–action formulations, while further improving performance on real-world tasks and substantially reducing action-side computation under unified profiling. These results highlight the world–action interface as a key design problem in WAMs, emphasizing the importance of selecting, compressing, and organizing control-relevant world representations for action prediction.

AI USE STATEMENT

Generative AI tools were used to assist with translation, language polishing, and improving the clarity and organization of the manuscript. All AI-assisted content was reviewed and revised by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training configuration, evaluation protocols, and efficiency measurement procedures in the appendix. The source code and model checkpoints will be publicly released upon acceptance.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A Unified Latent Action World Model. arXiv preprint arXiv:2512.13030, 2025. URL https://arxiv.org/abs/2512.13030.

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. arXiv preprint arXiv:2410.24164, 2024. URL https://arxiv.org/abs/2410.24164v1.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. arXiv preprint arXiv:2307.15818, 2023. URL https://arxiv.org/abs/2307.15818.

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. arXiv preprint arXiv:2505.06111, 2025. URL https://arxiv.org/abs/2505.06111.

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. arXiv preprint arXiv:2506.21539, 2025. URL https://arxiv.org/abs/2506.21539.

Zhirui Fang, Qingchi Yu, Ziyang Chen, Longfei Li, Haoran Ma, Keru Zhou, Xinrun Xu, Samith Va, Yuxuan Hu, Peixuan Song, Qiang Du, Bin Qian, Yongkang Deng, Xin Li, Yezhen Wang, Zhe Li, Hao Luo, Shuyan Li, Ziwei Wang, Weijian Deng, and Xiu Li. EMERGE-Policy: A Robot Mind Emerges Beyond a Single Policy. *arXiv preprint arXiv:2608.29896*, 2026. URL https://emerge-policy.github.io/EMERGE-Policy/.

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. arXiv preprint arXiv:2510.13626, 2025. URL https://arxiv.org/abs/2510.13626.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. NORA: A Small Open-Sourced Generalist Vision Language Action Model for Embodied Tasks. arXiv preprint arXiv:2504.19854, 2025. URL https://arxiv.org/abs/2504.19854.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv preprint arXiv:2406.09246, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. arXiv preprint arXiv:2502.19645, 2025. URL https://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. arXiv preprint arXiv:2601.16163, 2026. URL https://arxiv.org/abs/2601.16163.

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. arXiv preprint arXiv:2601.21998, 2026. URL https://arxiv.org/abs/2601.21998.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. arXiv preprint arXiv:2306.03310, 2023. URL https://arxiv.org/abs/2306.03310.

Guanxing Lu, Baoxiong Jia, Puhao Li, Yixin Chen, Ziwei Wang, Yansong Tang, and Siyuan Huang. GWM: Towards Scalable Gaussian World Models for Robotic Manipulation. arXiv preprint arXiv:2508.17600, 2025. URL https://arxiv.org/abs/2508.17600.

Hao Luo, Ye Wang, Wanpeng Zhang, Sipeng Zheng, Ziheng Xi, Chaoyi Xu, Haiweng Xu, Haoqi Yuan, Chi Zhang, Yiqing Wang, Yicheng Feng, and Zongqing Lu. Being-H0.5: Scaling Human-Centric Robot Learning for Cross-Embodiment Generalization. arXiv preprint arXiv:2601.12993, 2026. URL https://arxiv.org/abs/2601.12993.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, Wenbo Cui, Senmao Qi, Shuo Wang, Yixin Zheng, Mi Yan, Xuesong Shi, Haoran Li, Dongbin Zhao, Ming-Yu Liu, Zhizheng Zhang, Li Yi, Yizhou Wang, and He Wang. LDA-1B: Scaling Latent Dynamics Action Model via Universal Embodied Data Ingestion. arXiv preprint arXiv:2602.12215, 2026. URL https://arxiv.org/abs/2602.12215.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. DiT4DiT: Jointly Modeling Video Dynamics and Actions for Generalizable Robot Control. arXiv preprint arXiv:2603.10448, 2026. URL https://arxiv.org/abs/2603.10448.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. arXiv preprint arXiv:2406.02523, 2024. URL https://arxiv.org/abs/2406.02523.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. arXiv preprint arXiv:2212.09748, 2023. URL https://arxiv.org/abs/2212.09748v2.

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. arXiv preprint arXiv:2501.09747, 2025. URL https://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. arXiv preprint arXiv:2504.16054, 2025. URL https://arxiv.org/abs/2504.16054.

Jingjing Qian, Boyao Han, Chen Shi, Lei Xiao, Long Yang, Shaoshuai Shi, and Li Jiang. GeoPredict: Leveraging Predictive Kinematics and 3D Gaussian Geometry for Precise VLA Manipulation. arXiv preprint arXiv:2512.16811, 2026. URL https://arxiv.org/abs/2512.16811.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model. arXiv preprint arXiv:2509.09372, 2025. URL https://arxiv.org/abs/2509.09372.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, MingMing Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, Yingjie Cai, Xu Yan, Guanyi Zhao, Bingbing Liu, and Haoang Li. Robust-WAM: Bridging Generative Pretraining and Semantic Foresight in World-Action Models. arXiv preprint arXiv:2608.05903, 2026. URL https://arxiv.org/abs/2608.05903.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. arXiv preprint arXiv:2602.15922, 2026. URL https://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? arXiv preprint arXiv:2603.16666, 2026. URL https://arxiv.org/abs/2603.16666v2.

Chushan Zhang, Jinguang Tong, Xuesong Li, Yikai Wang, and Hongdong Li. Keep the Future, Drop the Rollout: RIFT for World Action Models. arXiv preprint arXiv:2608.11521, 2026a. URL https://arxiv.org/abs/2608.11521v2.

Jinhao Zhang, Zhexuan Zhou, Huizhe Li, Yichen Lai, Wenlong Xia, Haoming Song, Youmin Gong, and Jie Mei. Hyper-DP3: Frequency-Aware Right-Sizing of 3D Diffusion Policies for Visuomotor Control. arXiv preprint arXiv:2605.01581, 2026b. URL https://arxiv.org/abs/2605.01581v4.

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. arXiv preprint arXiv:2603.22078, 2026c. URL https://arxiv.org/abs/2603.22078v5.

Zijian Zhang, Yuqing Jiang, Qian Cheng, Xiaofan Li, Si Liu, Ding Zhao, Ping Luo, Weitao Zhou, and Haibao Yu. GaussianDream: A Feed-Forward 3D Gaussian World Model for Robotic Manipulation. arXiv preprint arXiv:2605.20752, 2026d. URL https://arxiv.org/abs/2605.20752v2.

A World-Feature Intervention Details

A.1 Intervention Setup

World-feature intervention protocol. Fast-WAM is our BS128 checkpoint at step 21,700; IDM and Joint use the FluxVLA and m1ku2 checkpoint files identified below. For each checkpoint, we compare Full, Selected, and Random using matched tasks and initial states.

Table A1: Intervention configurations across three WAM formulations.

	Fast-WAM	Fast-WAM-IDM	Fast-WAM-Joint
Checkpoint source	Local BS128	FluxVLA	m1ku2
Checkpoint step	21,700	21,700	40,000
World positions	98 current	294 current/future	294 current/future
Closed-loop retained positions	25/98	75/294	75/294
Offline observations (tasks)	8 (8)	80 (40)	80 (40)
Offline random orders	3	5	5

World positions and interventions. Each observation contains external- and wrist-camera views resized to 224×224 and concatenated horizontally. Each latent temporal group contains a 7×14 grid of 98 world positions. Fast-WAM exposes the current group only; IDM and Joint generate nine video frames, including the current frame, represented by three temporal groups. The future rows in the figures correspond to groups associated with frames 1–4 and 5–8, with generated frames 4 and 8 used as illustrative backgrounds. Each position represents a contextual world feature. Sensitivity replacement substitutes the complete 24-head K/V vectors at one position with their spatial means within the same layer and temporal group, including both cameras and the replaced position in that mean. Keys have already undergone normalization and rotary positional encoding. The same position is replaced jointly across all 30 layers and ten action-denoising steps. Retention applies a Boolean mask to excluded world-key columns only in action-query rows, preserving all 32 action keys, world generation, and the language and proprioception pathways.

Generated futures and replay. IDM generates video for ten denoising steps with the current latent slice clamped at each step, then reads the final video at timestep zero to obtain a fixed 30-layer cache for ten action steps. Joint first records the native ten-step joint pass. Within each replan, action interventions replay each action step using its corresponding recorded, unmodified world K/V cache. Both video and action schedulers use the native continuous flow-matching schedule with inference shift 5 and 1,000 training-time units. At the next replan, each condition constructs new world representations from its own observation.

A.2 Offline Sensitivity and Retention

Offline sensitivity and observations. Sensitivity is the RMSE between the replaced-feature and full-access final action predictions under the same observation and action-noise draw, averaged over seven normalized action components and valid, non-padding horizon indices. Predictions have horizon 32 and are compared at identical indices without temporal alignment or denormalization. Heatmaps average four per-draw sensitivities. Offline ranking and retention evaluation use the same LIBERO demonstration observations and world conditions. Ranking uses action-noise draw 0 with stable descending sorting, and retention is evaluated with draws 1–3. Fast-WAM uses eight observations, two per suite. IDM and Joint extend this set to 80 observations, with two observations from distinct demonstration episodes per task. The parent 256-observation manifest balances suites/tasks, uses distinct episodes, and samples one transition uniformly per episode. For IDM/Joint, each observation's generated world trajectory is shared across the four action draws; video generation and action-noise sampling use separate random streams. Sensitivity comparisons are made within each checkpoint, with model-specific heatmap scales. The Fast-WAM maximum uses all eight observations, whereas IDM/Joint maxima use the four displayed observations.

Offline retention. Random subsets are prefixes of uniform permutations sampled without replacement, with three random orders per observation for Fast-WAM and five for IDM/Joint. Masks

are nested across budgets within an order and reused across evaluation draws. Budgets are $\{0, 5, 10, 25, 49, 74, 98\}$ for Fast-WAM and $\{0, 15, 30, 75, 147, 222, 294\}$ for IDM/Joint. The 75-position subsets are chosen jointly from all 294 positions without per-frame or per-camera quotas. For each observation, we average RMSE over the three evaluation draws and all random orders, then average observations equally regardless of their valid action lengths.

A.3 Closed-Loop Evaluation

Closed-loop evaluation. Each model and each of Full, Selected, and Random is evaluated on task IDs 0–9 and saved initial-state indices 0–4 in Spatial, Object, Goal, and Long (`libero_10`): 200 episodes per condition, 600 per model, and 1,800 total. Each episode uses 30 dummy settling actions, predicts 32 actions, executes ten, and replans. Spatial, Object, and Goal allow 400 policy actions; Long allows 700. Each checkpoint retains its recorded preprocessing, denormalization, and evaluator gripper conversion/binarization; no action ensembling is used. Success is the unmodified evaluator's `bool(done)`. Selected recomputes its joint position ranking on its own current observation at every replan. The selected set is shared across all layers, heads, and action steps within that replan. Random uses one deterministic mask sequence per episode, refreshed at each replan. The diagnostic ranking probe uses action-noise draws separate from those used for policy execution.

Numerical execution and paired evaluation. Fast-WAM uses batch-eight BF16 graphs for offline interventions and online ranking, and native batch-one closed-loop action execution. Native-versus-batched differences are approximately 0.00147–0.00210 action RMSE. IDM/Joint use batch one for formal interventions, rankings, and actions. Fast-WAM comparisons use verified matching starts. IDM/Joint use fresh environments for each condition, with matched physical initial states, fixture transforms, 30 settling actions, 31 settling states, and state/RGB at the first policy input.

Error bars in Figure 2 indicate 95% hierarchical bootstrap confidence intervals. Plotted intervals are pointwise percentile intervals from 10,000 hierarchical bootstrap draws, resampling tasks within each fixed suite and observations (offline) or initial states (closed loop) within tasks. Closed-loop resamples are paired across conditions and averaged equally across suites.

A.4 Full Diagnostic Results

Closed-loop results. Full/Selected/Random success counts are $(192, 190, 149)$ for Fast-WAM, $(193, 196, 173)$ for IDM, and $(199, 195, 187)$ for Joint, each out of 200. Selected-minus-Random point-estimate differences are 20.5, 11.5, and 4.0 pp, respectively; Selected-minus-Full differences are -1.0, 1.5, and -2.0 pp.

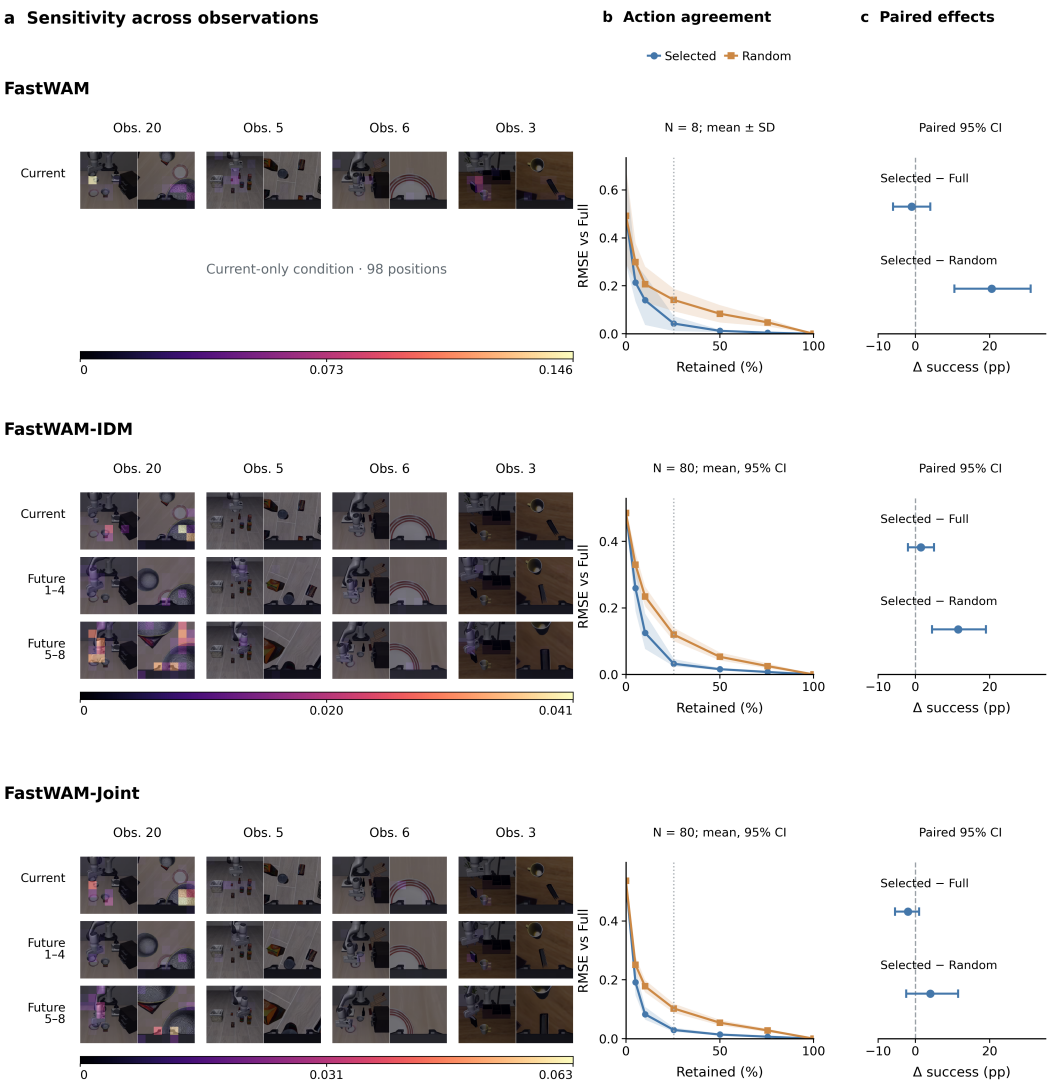

Figure A1: **Full world-feature intervention diagnostics.** Rows show Fast-WAM, Fast-WAM-IDM, and Fast-WAM-Joint. (a) Sensitivity maps across four observations. (b) Action RMSE relative to full-feature predictions under sensitivity-based or random retention; the dotted line marks 25.5% retention. (c) Paired success-rate differences over 200 episodes per model and condition. Shading and error bars show 95% bootstrap confidence intervals, except Fast-WAM shading in (b), which shows ± 1 sample SD.

B RAPID-WAM Implementation Details

B.1 Readout and Action Decoder

Readout networks. The default interface reads Wan layers $\{6, 12, 18, 24, 30\}$, using one-based indices. Under the LIBERO input configuration, each latent frame contributes 98 spatial tokens of width 3072 per tap. Hidden states $H_{\ell_k} \in \mathbb{R}^{n_k \times d_w}$ are projected to $P_k \in \mathbb{R}^{n_k \times d_c}$ with $d_c = 256$. The base queries, query MLP, cross-attention, refinement network, and output normalization are shared across taps; input normalizations, projections, and depth embeddings are depth-specific. The same spatial reader is applied independently to each latent frame in IDM and joint models.

The embedding e_ℓ is obtained by masked mean pooling of the conditioning context containing language and proprioceptive tokens, followed by layer normalization and a linear map to width

256. Proprioception is mapped by layer normalization and a two-layer GELU MLP to the same width. Each selected Wan depth has a separate normalization, a $3072 \to 256$ projection, and a learned 256-dimensional embedding. The shared query MLP uses layer normalization and widths $256 \to 512 \to 256$ with GELU. Four-head cross-attention produces eight query outputs per tap. The shared refinement network f_r uses pre-normalization and a $256 \to 1024 \to 256$ GELU MLP; its residual sum with the attention outputs is normalized before averaging across queries. The global network f_g applies layer normalization and a $256 \to 256 \to 256$ GELU MLP to the equally weighted mean of the original layer conditions plus the language and proprioceptive conditions.

Action blocks. The default ActionDiM uses $D = 8$ blocks, width $d_a = 256$, and action horizon $T = 32$. A linear action embedding and learned action-position embeddings map A_τ to $X^{(0)} \in \mathbb{R}^{T\times d_a}$. A sinusoidal timestep embedding followed by an MLP produces $e_\tau \in \mathbb{R}^{d_a}$. For the five-tap model, the routing in Equation 4 is $\pi = (1, 1, 2, 2, 3, 3, 4, 5)$.

Each block concatenates its local, global, and time conditions. Its condition MLP applies layer normalization and a $768 \to 256 \to 256$ SiLU MLP, followed by a separate linear map to the six modulation vectors. The temporal and channel MLPs use GELU and widths $32 \to 64 \to 32$ and $256 \to 1024 \to 256$, respectively; dropout is zero. Temporal and channel layer normalizations, condition MLPs, and modulation projections are distinct for each block.

The modulation output is $m_j = (\beta_j^T, \gamma_j^T, \alpha_j^T, \beta_j^C, \gamma_j^C, \alpha_j^C)$, with shift, scale, and residual-gate vectors for each branch. In action-only decoding these vectors are broadcast across action positions; in IDM and joint decoding, G_j acts on the position-dependent local conditions. Define $\mathcal{A}_j^b(X) = (1 + \gamma_j^b) \odot \mathrm{LN}_j^b(X) + \beta_j^b$ for $b \in \{T, C\}$. The block updates are

$$\begin{aligned} \widetilde{X}^{(j)} &= X^{(j-1)} + \alpha_j^T \odot \mathcal{T}_j\Big(\mathcal{A}_j^T(X^{(j-1)})\Big), \\ X^{(j)} &= \widetilde{X}^{(j)} + \alpha_j^C \odot \mathcal{M}_j\Big(\mathcal{A}_j^C(\widetilde{X}^{(j)})\Big). \end{aligned} \tag{6}$$

The temporal MLP $\mathcal{T}_j$ mixes the T action positions independently for each channel; the channel MLP $\mathcal{M}_j$ acts on each position independently. A final normalization and linear head output the action flow field. Action-time embeddings and block modulation are recomputed at every action sampling step, including when the interface conditions are reused.

B.2 Flow-Matching Schedule

Let $\phi_s(u) = su/(1 + (s - 1)u)$, with default action shift $s = 5$. Training samples $u \sim \mathcal{U}(0, 1)$ and sets $\tau = \phi_s(u)$. The time embedding uses the numerical timestep 1000τ. The loss weight is proportional to $\exp[-2(\tau - 1/2)^2] - \exp(-1/2)$, normalized by its mean on the scheduler's 1,000-point shifted grid. Padded action positions are excluded before averaging the squared error. Inference uses $\tau_r = \phi_s(1 - r/N)$ for $r = 0, \ldots, N$, and the Euler update $A_{\tau_{r+1}} = A_{\tau_r} + (\tau_{r+1} - \tau_r)v_\theta(A_{\tau_r}, \tau_r; C, g)$. This schedule decreases the noise level from one to zero.

The joint configuration uses action shift $s = 1$; formulation-specific settings are given in Appendix B.4.

B.3 Temporal Conditioning for IDM and Joint

Temporal readout and routing. For IDM and joint models, the shared spatial reader produces one summary for each selected depth and latent frame. At each depth k, a temporal router aggregates the future summaries according to the task and action position, producing a residual $r_{k,t}$ for action position t. The local control condition becomes

$$\widetilde{c}_{k,t} = c_k^{\mathrm{cur}} + \gamma_{k,t} r_{k,t}. \tag{7}$$

Here, c_k^{cur} is the summary of the current observation, $r_{k,t}$ collects future information relevant to action position t, and $\gamma_{k,t}$ gates its contribution. The decoder uses $\widetilde{c}_{\pi(j),t}$ as its local condition, while the global condition g is computed from current-frame summaries only.

The temporal router uses four heads of width 64. Its queries combine fixed sinusoidal action-position embeddings with the language and proprioceptive condition $e_\ell + e_p$, followed by layer normalization

and a bias-free linear map. Future summaries are normalized and linearly projected into keys, with learned future-position embeddings added to the keys. The values are the unchanged channel slices of the future summaries. Softmax attention over future latent positions produces $r_{k,t}$; the default router supports two future latent frames.

Future gates. In IDM, $\gamma_{k,t}$ is a learned scalar for each depth and action position, initialized to zero. In joint models, $\gamma_{k,t}(\sigma) = \gamma^0_{k,t} + [f_\gamma(e_{1000\sigma})]_{k,t}$ additionally depends on the current video noise level σ, where $e_{1000\sigma}$ is a 64-dimensional sinusoidal embedding of the native video timestep. The gate network uses a $64 \to 64 \to KT$ SiLU MLP with a bias-free output layer; both the base gate and final weight are initialized to zero. The gates are unconstrained scalars. Current-frame tokens attend only to the current frame, preserving the current observation anchor; video computation receives no action inputs.

B.4 Formulation-Specific Inference and Training

Action-only inference. In action-only inference, the world backbone processes only the current observation. The interface constructs C and g once per replan and reuses them throughout action denoising.

Inverse-dynamics inference. The model first generates future video latents. The spatial reader is then applied independently to the current and future representations obtained from the final video representation. The temporal router combines the resulting summaries using Equation 7. These conditions are constructed once after future generation and remain fixed during action denoising.

Joint world–action inference. World and action states evolve together. At every joint denoising step, RAPID-WAM applies the same spatial reader and temporal router to the current and evolving future representations, using the noise-dependent gate $\gamma_{k,t}(\sigma)$.

Formulation-specific training. IDM computes the video objective on a noised-video pass and the action objective on a separate conditioning-video pass. For each sample, the latter uses clean or noised demonstration video with probability 0.5 each, retaining the clean current frame. Joint training obtains action conditions from the noised-video branch, with video and action noise levels sampled independently. Both objectives use the scheduler weights in Appendix B.2. Action gradients pass through the spatial readout and temporal router into the backbone.

Formulation-specific settings. IDM uses video and action shifts of 5; Joint uses an action shift of 1. Common training settings and LIBERO/LIBERO-Plus evaluation protocols are given in Appendix C.

C Training and Evaluation Protocols

The training recipe below applies to our simulation experiments with RAPID-WAM and the Fast-WAM checkpoints used for efficiency and RoboCasa; cited benchmark entries retain their source protocols. Real-world training and evaluation details are provided in Appendix F. For each configuration, results are reported for a fixed trained policy.

C.1 Optimization

Our canonical RAPID-WAM and Fast-WAM runs start from pretrained Wan weights and jointly update the world backbone and action modules for 21,700 steps. Fast-WAM's action tower uses interpolated initialization; the RAPID-WAM readout and ActionDiM are newly initialized. Training uses AdamW with peak learning rate 10^{-4}, 5% warm-up, cosine decay to 10^{-6}, $\beta = (0.9, 0.95)$, weight decay 0.01, and gradient clipping at 1.0. The effective batch size is 128, using four processes, 16 samples per process, and two accumulation steps. We use BF16 mixed precision and equally weighted video and action objectives.

C.2 LIBERO AND LIBERO-PLUS

We mix the no-noops demonstrations from Spatial, Object, Goal, and Long (LIBERO-10) to train one policy. LIBERO average success is the mean of the four suite success rates. Two camera views are resized to 224×224 and concatenated horizontally. Actions and proprioception have 7 and 8 dimensions, respectively, with min–max normalization. The action horizon is 32; the policy executes ten actions before replanning, after an initial 30-step environment wait. Inference uses CFG scale 1, action shift 5, and ten action-network calls unless stated otherwise. LIBERO evaluation uses 50 rollouts per task. LIBERO-Plus uses the same trained policy and one rollout for each of 10,030 variants. Its camera, robot-state, language, lighting, background, noise, and layout groups contain 1,599, 1,550, 1,537, 1,142, 1,076, 1,601, and 1,525 instances. Overall success weights each rollout equally.

C.3 ROBOCASA

We train separately on the Human-50 success-only preprocessing, containing 1,199 trajectories from 24 tasks. Three 224×224 camera views are arranged into a 384×320 input. Action and state dimensions are 7 and 9. We use min–max normalization, deterministic environments, and flipped input images. The policy predicts 32 actions and executes 16 before replanning, with ten sampling steps. Evaluation uses split B and layout/style pairs $(1, 1)$, $(2, 2)$, $(4, 4)$, $(6, 9)$, and $(7, 10)$. Each model is evaluated on 24 tasks with 50 rollouts per task, totaling 1,200 rollouts per model.

C.4 BASELINE RESULT SOURCES

External baseline scores follow published papers, including subsequent benchmark evaluations, or official project and benchmark releases, and retain their source evaluation protocols. Fast-WAM's LIBERO and LIBERO-Plus entries in the main benchmark tables use published results, while its RoboCasa and real-world results and the checkpoints used for efficiency experiments come from our own runs. A comparison with our locally trained action-only Fast-WAM is provided in Appendix D.1.

Method references. The VLA baselines include OpenVLA (Kim et al., 2024), WorldVLA (Cen et al., 2025), NORA (Hung et al., 2025), UniVLA (Bu et al., 2025), π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), OpenVLA-OFT (Kim et al., 2025), and $\pi_{0.5}$ (Physical Intelligence et al., 2025). The WAM baselines include Motus (Bi et al., 2025), LingBot-VA (Li et al., 2026), DiT4DiT (Ma et al., 2026), LDA-1B (Lyu et al., 2026), Robust-WAM (Yan et al., 2026), and Fast-WAM (Yuan et al., 2026), including its IDM and Joint variants.

LIBERO (Table 1). Fast-WAM and several baselines follow Yuan et al. (2026); additional entries follow Ma et al. (2026) and the VLA-Adapter-Pro release (Wang et al., 2025). WorldVLA uses 512×512 inputs; NORA denotes fine-tuned NORA-Long; UniVLA uses full pretraining. π_0-FAST follows the LIBERO results collected by the LIBERO-Plus authors (Fei et al., 2025).

RoboCasa (Table 2). Grouping follows the policy backbone; VLA-based methods include geometry-augmented policies. Group scores follow Zhang et al. (2026d, Table 2), with GWM, GeoPredict, and Being-H0.5 scores taken from their original papers (Lu et al., 2025; Qian et al., 2026; Luo et al., 2026). Pick & Place, Doors/Drawers, and Others contain 8, 6, and 10 tasks. To use the same task-weighted averaging as our models and other benchmark methods, we recompute GaussianDream's overall success from its reported group scores with weights $8 : 6 : 10$, giving 53.8%. Its reported 54.8% matches the unweighted mean of the three published group scores.

LIBERO-Plus (Table 3). The Fast-WAM entry follows Zhang et al. (2026a). LingBot-VA follows Fang et al. (2026). UniVLA results follow Table 2 of the LIBERO-Plus paper (Fei et al., 2025), with an overall success rate of 42.9%. WorldVLA, NORA, and π_0-FAST follow the official LIBERO-Plus leaderboard (Fei et al., 2025). LDA-1B and Robust-WAM follow Yan et al. (2026). OpenVLA, OpenVLA-OFT, and π_0 results follow Fei et al. (2025, Table 2). $\pi_{0.5}$ results follow Zhang et al. (2026c, Table 4). Robust-WAM denotes its Fast-WAM-based variant.

D Additional Experimental Results

D.1 Full Results across WAM Formulations

Local action-only comparison. The main benchmark tables use published Fast-WAM success rates of 97.6% on LIBERO and 49.7% on LIBERO-Plus, which are higher than our local reproduction's 95.9% and 45.6%, respectively. Table A2 complements these published comparisons with results from our locally trained action-only models. Both models use the same LIBERO demonstrations, 21,700 optimization steps, an effective batch size of 128, and the AdamW schedule in Appendix C.1, and are evaluated using the protocols in Appendix C.2. RAPID-WAM achieves 98.8% and 70.7%, respectively, maintaining higher success on both benchmarks under these shared settings.

Table A2: Success rates (%) of locally trained action-only Fast-WAM and RAPID-WAM.

Method	LIBERO	LIBERO-Plus
Fast-WAM	95.9	45.6
RAPID-WAM	**98.8**	**70.7**

IDM results. On LIBERO, RAPID-WAM-IDM succeeds in 498, 500, 488, and 486 out of 500 episodes on Spatial, Object, Goal, and Long, totaling 1,972/2,000. On LIBERO-Plus it succeeds in 7,225/10,030 rollouts, giving 72.0%.

Table A3: LIBERO-Plus success counts and rates for RAPID-WAM-IDM, with one rollout per benchmark variant.

Perturbation	Successes / rollouts	Success (%)
Camera	1059 / 1599	66.2
Robot initial state	768 / 1550	49.5
Language	1371 / 1537	89.2
Lighting	1053 / 1142	92.2
Background texture	518 / 1076	48.1
Sensor noise	1269 / 1601	79.3
Object layout	1187 / 1525	77.8
Overall	7225 / 10030	72.0

Joint results. RAPID-WAM-Joint succeeds in 494, 499, 490, and 489 out of 500 episodes on Spatial, Object, Goal, and Long, respectively, totaling 1,972/2,000 (98.6%). On LIBERO-Plus, it succeeds in 7,247 of 10,030 rollouts, yielding 72.3% overall success.

Success breakdowns across formulations. Table A4 reports the full LIBERO suite and LIBERO-Plus perturbation-category results across formulations.

Table A4: Success rates (%) across action-only, inverse-dynamics (IDM), and joint world–action formulations. Bold marks the higher LIBERO average and LIBERO-Plus overall success within each formulation.

	LIBERO					LIBERO-Plus							
Method	Spatial	Object	Goal	Long	Avg.	Camera	Robot	Lang.	Light	BG	Noise	Layout	Overall
Fast-WAM	98.2	100.0	97.0	95.2	97.6	16.5	43.4	67.8	80.1	51.6	37.9	61.2	49.7
RAPID-WAM	99.0	99.8	98.4	97.8	**98.8**	64.2	47.3	90.4	90.4	44.9	78.1	77.2	**70.7**
Fast-WAM-IDM	98.8	97.8	97.8	97.6	98.0	46.2	69.1	94.2	92.8	56.4	61.8	81.5	71.4
RAPID-WAM-IDM	99.6	100.0	97.6	97.2	**98.6**	66.2	49.5	89.2	92.2	48.1	79.3	77.8	**72.0**
Fast-WAM-Joint	99.6	99.4	98.2	96.8	98.5	38.7	63.9	93.1	94.7	55.7	57.7	77.5	68.1
RAPID-WAM-Joint	98.8	99.8	98.0	97.8	**98.6**	66.2	50.2	89.9	92.7	50.8	77.3	77.7	**72.3**

D.2 Full Readout Ablations

Tables A5, A6, and A7 provide the full results underlying Figure 4(a,b). The shared default configuration uses the same evaluation across all three comparisons.

Readout depth. Five taps improve camera, language, noise, and layout success relative to the final-layer-only configuration, while robot-state, lighting, and background-texture success are lower.

Table A5: LIBERO and LIBERO-Plus success rates (%) across world-model readout depths. All configurations use eight queries per tap, an eight-block width-256 decoder, and $N = 10$; layer indices are one-based. Bold marks column-wise best values.

		LIBERO					LIBERO-Plus							
Taps	Layers	Spatial	Object	Goal	Long	Avg.	Camera	Robot	Lang.	Light	BG	Noise	Layout	Overall
1	$\{30\}$	**99.4**	**99.8**	**98.4**	97.6	**98.8**	59.4	**52.5**	82.0	92.1	**48.1**	70.5	70.4	67.7
3	$\{10, 20, 30\}$	98.8	99.6	**98.4**	95.8	98.2	62.5	49.8	87.2	91.9	39.9	75.1	**80.4**	70.0
5 (default)	$\{6, 12, 18, 24, 30\}$	99.0	**99.8**	**98.4**	**97.8**	**98.8**	64.2	47.3	**90.4**	90.4	44.9	**78.1**	77.2	70.7
10	$\{3, 6, \ldots, 30\}$	97.8	**99.8**	98.0	95.4	97.8	57.2	49.2	87.0	90.1	45.2	71.5	79.4	68.6
30	$\{1, 2, \ldots, 30\}$	97.8	**99.8**	97.6	95.0	97.6	**64.9**	48.9	89.7	**93.9**	44.5	73.5	78.9	**70.8**

Table A6: Success rates (%) for projected mean pooling and learned readout. Bold marks the higher LIBERO average and LIBERO-Plus overall success.

	LIBERO					LIBERO-Plus							
Method	Spatial	Object	Goal	Long	Avg.	Camera	Robot	Lang.	Light	BG	Noise	Layout	Overall
Projected mean pooling	97.2	98.8	98.4	95.6	97.5	40.1	48.5	82.8	90.1	45.2	62.4	66.7	61.8
Learned readout	99.0	99.8	98.4	97.8	**98.8**	64.2	47.3	90.4	90.4	44.9	78.1	77.2	**70.7**

Query count. Each tap outputs a 256-dimensional condition for every query count.

Table A7: Query-count ablation with five readout taps and $N = 10$. Success rates are percentages; bold marks column-wise best values.

	LIBERO					LIBERO-Plus							
Q	Spatial	Object	Goal	Long	Avg.	Camera	Robot	Lang.	Light	BG	Noise	Layout	Overall
2	97.0	**100.0**	98.2	97.0	98.1	60.5	43.7	84.4	91.5	43.0	76.0	75.8	68.0
4	98.0	99.8	**98.6**	97.2	98.4	65.7	46.8	85.4	**94.3**	**45.6**	76.8	77.8	70.5
8 (default)	**99.0**	99.8	98.4	**97.8**	**98.8**	64.2	47.3	**90.4**	90.4	44.9	78.1	77.2	70.7
16	98.2	99.6	96.6	96.4	97.7	**69.5**	**51.3**	86.1	92.9	42.8	**82.7**	**79.7**	**72.7**

Reader attention and retention. We analyze the default action-only RAPID-WAM model. The attention score of each spatial position is obtained by averaging attention weights over four heads, eight queries, and five readout depths. To measure action sensitivity, we replace the same spatial position at all five depths with the corresponding depth's spatial mean, recompute the interface conditions, and generate actions. Sensitivity is the RMSE between the original and intervened normalized 32×7 action chunks, averaged over four paired initial action-noise draws. Figure 4(c) shows two observations fixed before the analysis. Heatmap color and opacity use square-root scaling, with a shared scale across views and examples within each metric.

We compare full access, high-attention retention, and random retention on 40 LIBERO tasks with five matched initial states per task, totaling 200 episodes per condition. All three conditions use the same model and are evaluated on the same task and initial-state set. At each replan, high-attention retention selects the top 25 positions according to reader attention on the current observation, while random retention uniformly samples 25 distinct positions from the 98 available positions. The retained set is shared across all five readout depths, after which the interface conditions and actions are recomputed. The world backbone still computes its full representations; retention only restricts the reader's access to spatial positions. The success axis in Figure 4(d) spans 90–100%.

Table A8: Closed-loop LIBERO results under full access and attention-based or random retention. Entries report successes/episodes (success rate).

Suite	Full	High-attention	Random
Spatial	50/50 (100.0%)	50/50 (100.0%)	49/50 (98.0%)
Object	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)
Goal	50/50 (100.0%)	49/50 (98.0%)	46/50 (92.0%)
Long	48/50 (96.0%)	49/50 (98.0%)	42/50 (84.0%)
Overall	198/200 (99.0%)	198/200 (99.0%)	187/200 (93.5%)

On this evaluation set, high-attention retention matches the overall success rate of full access and exceeds random retention by 5.5 percentage points, with the gains over random retention mainly coming from Goal and Long (Table A8).

D.3 Qualitative LIBERO-Plus Results

Figure A2 illustrates selected paired executions under camera-viewpoint changes, sensor noise, and language paraphrasing. Paired executions use our locally trained Fast-WAM and RAPID-WAM checkpoints at 21,700 updates. Each pair shares the recorded initial state, perturbation instance and evaluation configuration, including ten action-network calls per chunk, an action horizon of 32, and replanning every ten actions.

Panel (c) uses the instruction "could you do me a favor pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate near the sink."

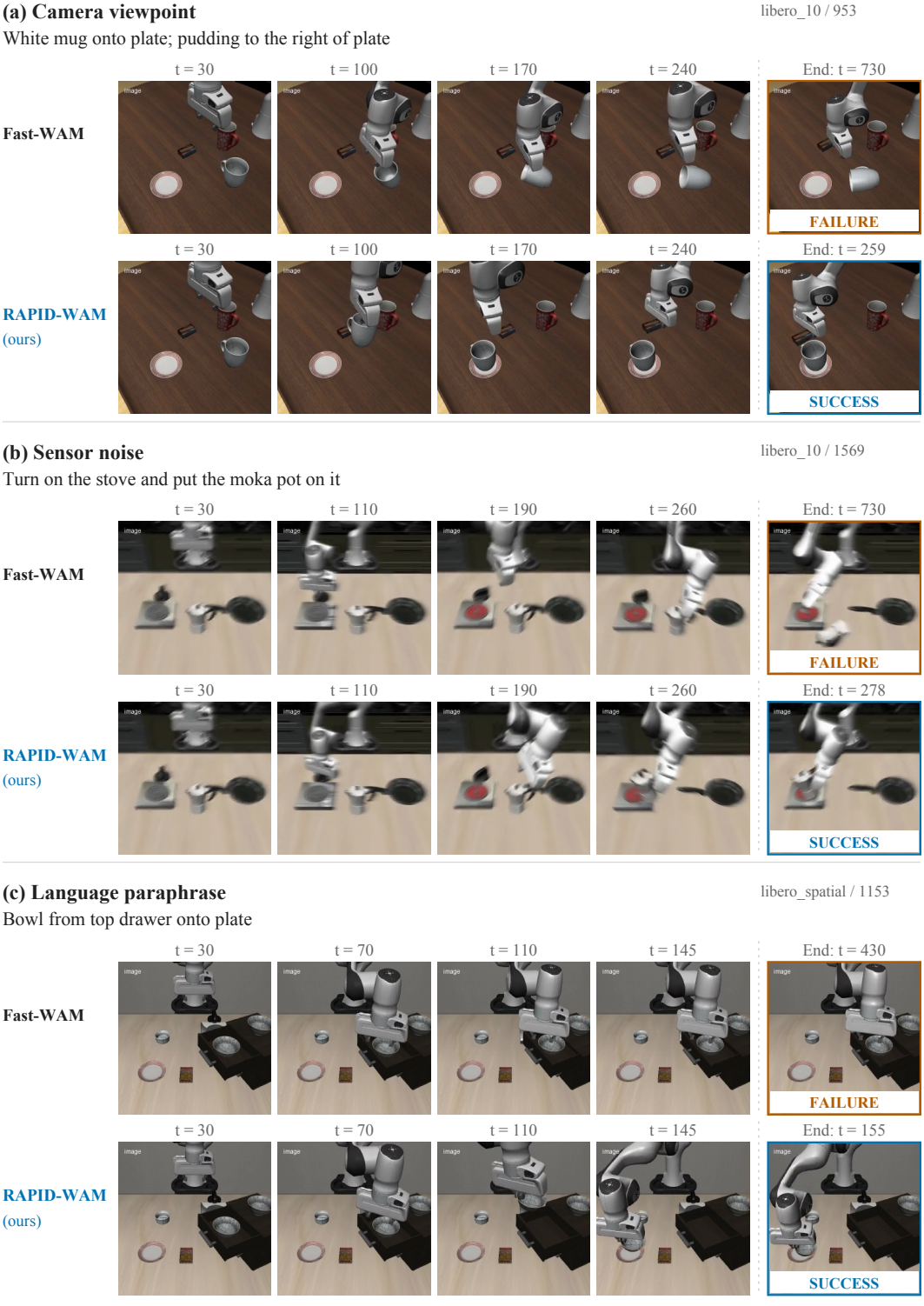

Figure A2: **Selected paired LIBERO-Plus executions** using locally trained Fast-WAM (top) and RAPID-WAM (bottom). Columns show matched simulator observations, with the final column showing each episode's terminal state. Cases are selected for RAPID-WAM success and Fast-WAM failure. Timestep labels denote environment time.

E Efficiency Accounting and Measurement Protocol

Decoder parameter counts exclude interface modules. Action-stage GFLOPs are reported per action chunk and include interface computation. Action-generation and policy latencies are medians for the action stage and full policy call, respectively.

Interface computation. For an action-only chunk using N action-network calls, we count

$$F_{\text{interface}}(N) = F_{\text{once}} + N F_{\text{inject}}. \tag{8}$$

The first term constructs conditions from already-computed world features; the second reads and injects them at each action step. In GFLOPs, the measured operator accounting gives

$$F_{\text{interface}}^{\text{Fast-WAM}}(N) = 25.726281600\, N, \tag{9}$$

$$F_{\text{interface}}^{\text{RAPID-WAM}}(N) = 0.960611328 + 0.011014144\, N. \tag{10}$$

Fast-WAM includes its full mixed-attention branch and associated modulation, gating, and residual operations, including action self-interaction. Native world K/V generation belongs to the backbone. RAPID-WAM includes the once-per-observation readout and each step's condition MLP, AdaLN projection, modulation, gating, and residual operations. Both exclude the VAE, world backbone, shared input preparation, independent timestep encoding, action FFN/mixer core, input/output heads, and scheduler from this interface scope. Fast-WAM's separate language/proprioception cross-attention and FFN-branch modulation are also outside this scope.

Action stage and parameters. Action-stage FLOPs include interface computation, all action-network calls, and scheduler updates after world condition sources are ready. At $N = 10$, interface and action-stage FLOPs are reduced by $240.26\times$ and $268.59\times$, respectively. Full-policy FLOPs, including the VAE and world backbone, fall from 2546.198 to 1494.913 GFLOPs, a $1.70\times$ ratio. Decoder parameter counts are 1,020,886,023 for Fast-WAM and 9,661,959 for RAPID-WAM; the latter's independent readout adds 6,013,712 parameters. All FLOP counts use 1 MAC = 2 FLOPs and count executed matrix products, convolutions, attention products, and separately dispatched arithmetic. Fused normalization, activations, softmax, reductions, bias, trigonometric operations, and some small operators are not separately counted.

Delivered representation size. We count scalar elements in the world-derived conditions used by one action-decoder evaluation with the LIBERO inputs in Table 5: layerwise world K/V for Fast-WAM and the final conditions C and g for RAPID-WAM, after spatial readout and any temporal routing. We exclude the batch dimension, interface intermediates, and decoder-generated modulation, and count each condition tensor once even when it is reused across decoder blocks or action-sampling steps. The eight query outputs per tap are intermediate activations and are pooled before delivery. Fast-WAM's separate language/proprioception inputs are outside this world-conditioning scope; C and g already incorporate these inputs in RAPID-WAM.

In the action-only formulation, the current observation supplies 98 world tokens. Each of the 30 Fast-WAM layers exposes a key and a value tensor with 24 heads of width 128, giving $30 \times 2 \times 98 \times 24 \times 128 = 18{,}063{,}360$ scalar elements. Here, the factor of two counts K and V. RAPID-WAM supplies $C \in \mathbb{R}^{5\times 256}$ and $g \in \mathbb{R}^{256}$, giving $5 \times 256 + 256 = 1{,}536$ elements, an $11{,}760\times$ reduction.

For IDM and joint world–action, the evaluated nine-frame video contains one current and eight future RGB time positions. VAE temporal compression yields three latent time positions, each with 98 spatial tokens, for 294 world tokens in total. Fast-WAM therefore exposes $30 \times 2 \times 294 \times 24 \times 128 = 54{,}190{,}080$ elements per action-decoder evaluation. In RAPID-WAM, the temporal router constructs conditions for each of the 32 action positions, yielding $C \in \mathbb{R}^{5\times 32\times 256}$; the global condition remains $g \in \mathbb{R}^{256}$ and is formed from the current observation. The delivered size is therefore $5 \times 32 \times 256 + 256 = 41{,}216$ elements, a reduction of approximately $1{,}314.78\times$. These action-position-dependent conditions include the routed future information and are counted at their final shape.

Action-only conditions are constructed once per replan; IDM conditions are constructed after future generation. Both are reused throughout action sampling. Joint conditions are refreshed at each denoising step, and its reported size is per step. We do not multiply the representation sizes in Table 5 by the $N = 10$ action-network calls.

Timing. We replay the same captured LIBERO observation on the same RTX 5880 with batch one, evaluation mode, no gradients, BF16 with the models' existing FP32 layers, and eager execution. Both model families use 32-action chunks and $N = 10$ action-network calls unless stated otherwise. Each of three rounds uses 20 warm-up calls followed by 100 timed calls. We compute p50 over all 300 measurements. CUDA-event timing covers the full policy call, including VAE encoding, the world backbone, action generation, and output transfer, but excludes image acquisition, external preprocessing, text encoding, and simulator execution. Text features are precomputed. The VAE, world backbone, and visual readout are recomputed for every observation, with conditions reused only within the chunk.

F Real-World Experimental Details

Hardware and observations. The real-world experiments use a bimanual robot platform equipped with two AgileX Piper arms. Only one arm is enabled for each task; the other remains powered off and receives no control commands. Visual observations are captured by one Intel RealSense D455 camera providing a global view and two ORBBEC Dabai wrist cameras mounted on the left and right arms, respectively. Both methods use the same two visual inputs, the global D455 view and the wrist view of the active arm. Policies run on an NVIDIA RTX 4090 GPU for online action inference. The model outputs a 7-dimensional action for the active arm, including six joint positions and one gripper command.

Data and training. We collect expert demonstrations using a leader–follower teleoperation system, with 150 trajectories per task. For each method, we train a separate policy for each of the two tasks using only the 150 demonstrations for that task. Fast-WAM and RAPID-WAM use exactly the same demonstration data for a given task.

Tasks. We separately evaluate two independent single-arm manipulation tasks, as shown in Figure A3.

- **Cup placement:** the left arm grasps a cup, places it on a plate, and releases it.
- **Plush toy placement:** The cup is already on the plate at initialization. The right arm grasps an eggplant-shaped plush toy, places it inside the cup, and releases it. This task is evaluated independently of cup placement.

Evaluation protocol. Each method is evaluated with 40 physical rollouts per task, totaling 80 rollouts across the two tasks. Both methods use the same initial conditions for each task and are tested alternately during evaluation. We evaluate Fast-WAM under the same protocol, with 18/40 and 13/40 successes on cup placement and plush toy placement, respectively; RAPID-WAM achieves 22/40 and 16/40 successes, respectively. Table 4 reports the corresponding success rates for each task and the average success rate across the two tasks.

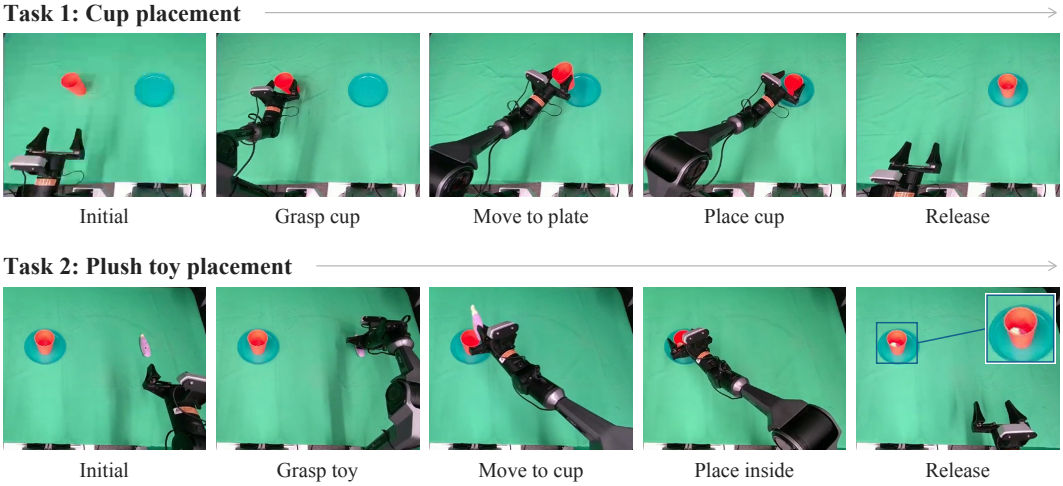

Figure A3: **Real-world task illustrations.** The top row shows the left arm grasping a cup, placing it on a plate, and releasing it. The bottom row shows the right arm placing an eggplant-shaped plush toy into a cup already positioned on the plate. Each row contains five chronological frames; the two rows are not temporally aligned. The inset enlarges the cup from the final frame of the bottom row to highlight the toy inside. Quantitative results for the two independently evaluated tasks are reported in Table 4.