LONG-WAM: SCALING THE CONTEXT OF WORLD-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time robot control demands enough visual history to infer motion and task progress, but processing that history can delay action. We present **Long-WAM**, a model–system framework for scaling the context of causal world-action models under real-time control constraints. Our central finding is that access to history is not the same as using it: longer histories pay off far more when the video foundation is pretrained autoregressively (AR). We first learn causal prediction from robot and egocentric videos without action labels, then preserve this history-to-future structure during world-action adaptation. On RoboCasa GR-1, increasing context from 0.0 to 19.2 seconds raises success from 63.3% to 78.7%, whereas a bidirectionally pretrained initialization shows no net gain; robot-domain AR pretraining further raises peak success on GR-1 and LIBERO-Long. Long-WAM also achieves the best results among compared methods on LIBERO-Long, RoboTwin 2.0, and DOMINO. Streaming observation encoding, asynchronous execution, and hardware-specific acceleration enable deployment on RTX 5090, DGX Spark, and Jetson AGX Thor without dropping future prediction; on RTX 5090, each action chunk, including future-video latent prediction, takes 107.4 ms. Real-time deployment on Unitree G1 and YAM supports dynamic and long-horizon manipulation, including 95% success on dynamic cup stacking, where $\pi_{0.5}$ and Fast-WAM succeed in none of 20 trials. As a memory-informed executor, Long-WAM also complements higher-level planning in composite tasks.

1 INTRODUCTION

Fast robot control requires understanding how the scene is changing. A single image can reveal an object's position but leave its motion and interaction progress ambiguous. World-action models (WAMs) bring video prediction to closed-loop control (Bi et al., 2025; Ye et al., 2026b; Li et al., 2026a), making visual history a natural resource for action. Yet processing more history can delay the response it is meant to improve. We therefore ask: *how does WAM control scale with visual context, and how can those gains be retained under real-time constraints?*

Recent WAMs retain history through causal caches and persistent or selected memories (Yang et al., 2026; Sun et al., 2026a; Wang et al., 2026a; Su et al., 2026). Yet *access to history is distinct from learning to predict from it*. DreamZero (Ye et al., 2026b) and LingBot-VA (Li et al., 2026a) adapt bidirectionally pretrained video generators to causal video–action prediction, whereas LingBot-VA 2.0 (Zhang et al., 2026a) jointly pretrains causal video and learned latent actions. We investigate a complementary route: first learn autoregressive (AR) video prediction, then adapt to actions while preserving its history-to-future structure. Appendix B discusses related work.

We introduce **Long-WAM** (Figure 1), a model–system framework for context scaling in real-time robot control. Our central finding is that *the value of longer context depends on how the video foundation is pretrained* (Figure 5). On GR-1, both AR-pretrained foundations gain from extending context from 0.0 to 19.2 seconds, whereas a bidirectional initialization does not; the robot-domain AR model's lead over it grows from 3.3 points without history to 17.1 points at 19.2 seconds. AR pretraining already learns the causal temporal factorization that WAM adaptation retains, giving the model a basis for using history rather than merely accessing it. Our LongLive2.0-Robot foundation learns from roughly 10,000 window-equivalent hours of robot and egocentric video; we preserve its

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

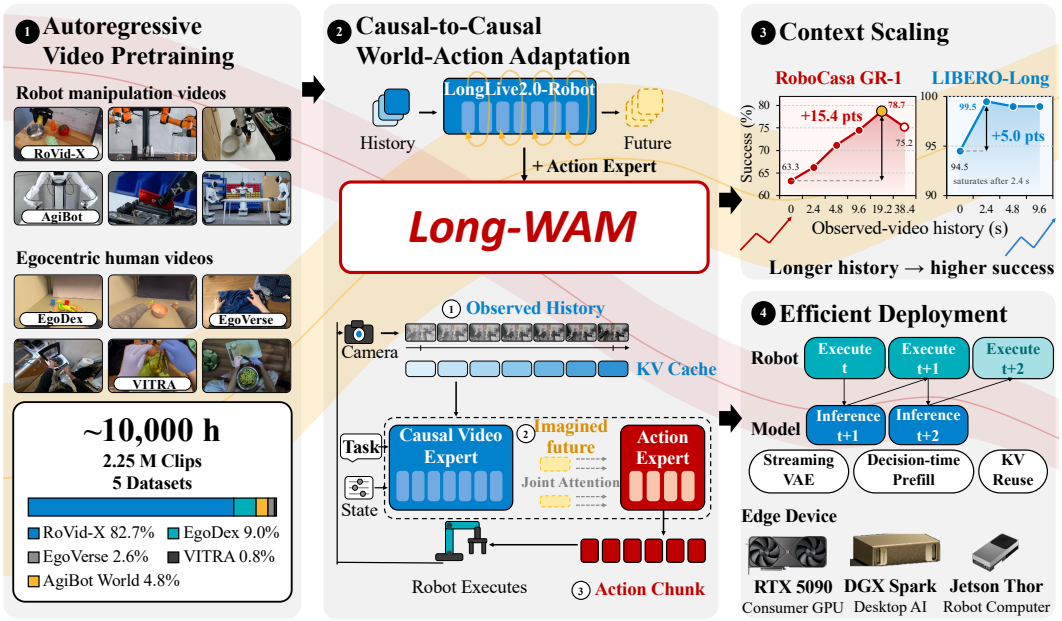

Figure 1: **Overview of Long-WAM.** Causal pretraining on robot and egocentric videos provides the predictive foundation. Observed history and partially denoised future latents condition action-chunk generation. We investigate how closed-loop performance scales with observed-video context and combine inference acceleration with asynchronous execution for real-time control on edge devices.

causal structure during action adaptation, and robot-domain pretraining further raises peak success on both LIBERO-Long and GR-1.

Longer context helps only if the controller still responds in time, so we co-design asynchronous execution and edge acceleration while retaining future prediction. Predictive conditioning from the AR video foundation supports smooth action handoffs without blending or prefix guidance; streaming VAE encodes incoming observations while the robot executes, reducing post-trigger computation; and NVFP4 quantization with device-specific kernel tuning accelerates the video–action pipeline on RTX 5090, DGX Spark, and Jetson AGX Thor. With four denoising steps per expert, the optimized pipeline takes 107.4 ms per action chunk on RTX 5090, including the full observation VAE computation, a 3.3× speedup over BF16 eager execution (Section 3, Table 7).

On RoboCasa (Nasiriany et al., 2024) GR-1 Tabletop (NVIDIA GEAR, 2025), increasing the history window from 0.0 to 19.2 seconds raises success from 63.3% to 78.7% (+15.4 points). On LIBERO-Long (Liu et al., 2023), 2.4 seconds of history improves success from 94.5% to 99.5%. Long-WAM also achieves 94.4% average success on RoboTwin 2.0 (Chen et al., 2025a) and 34.9% success on DOMINO (Fang et al., 2026), the highest among compared methods. On a Unitree G1, Long-WAM grasps cups from a conveyor moving at 7.5 cm/s in 90% of trials and stacks moving cups in 95%, settings in which neither $\pi_{0.5}$ (Physical Intelligence et al., 2025) nor Fast-WAM (Yuan et al., 2026b) succeeds once. On YAM, it completes tasks lasting over 40 seconds on average with 81.7% success, extending the evaluation from rapid interception to sustained execution. Long-WAM also serves as the executor of a hierarchical system on RoboCasa365 (Nasiriany et al., 2026): pairing its unchanged checkpoint with GPT-6 Astra raises overall success from 31.4% to 54.4%, versus 25.2% for the planner alone. Planner augmentation yields a larger gain for Long-WAM (+23.0 points) than for $\pi_{0.5}$ (+13.6; Appendix F), suggesting that stronger execution lets planning pay off more. Together, these results suggest that context is a resource for generative control whose value depends on both predictive pretraining and timely execution.

2 Scaling the Context of WAMs from AR Video Generation

Long-WAM learns a causal predictive foundation from robot video (Section 2.1), couples it to action generation (Section 2.2), and scales the observed history available for control (Section 2.3).

2.1 LONGLIVE2.0-ROBOT: ROBOT-DOMAIN AR VIDEO PRETRAINING

Data and initialization. LongLive2.0-Robot continues training from LongLive-2.0's 16-second AR checkpoint (Chen et al., 2026b) on approximately 10,000 window-equivalent hours from RoVid-X (Deng et al., 2026), AgiBot World (AgiBot-World-Contributors et al., 2025), EgoDex (Hoque et al., 2026), EgoVerse (Punamiya et al., 2026), and VITRA (Li et al., 2025). Because supervision is video-only, the model can learn from multiple embodiments without requiring a shared action space. Appendix C.1 details the data accounting.

Teacher-forcing training. We pretrain on robot-video sequences up to 30 seconds long, exposing the model to extended motion and interaction histories. To support this temporal span, we adopt LongLive-2.0's sequence-parallel AR training (Chen et al., 2026b), which shards long sequences across GPUs. Following the teacher-forcing formulation for AR video generation (Zhou et al., 2025), block-causal attention supervises every noisy chunk from its ground-truth prefix in parallel, while the conditioning image stays clean and outside the loss. We retain LongLive-2.0's error recycling, derived from SVI (Li et al., 2026b). For a clean target chunk z_i, let $\bar{z}_i$, $\bar{\epsilon}_i$, and $h_{<i}$ denote the target latent, sampled Gaussian noise, and preceding ground-truth context after optional buffered-error perturbations. The noisy input and teacher-forcing objective are

$$x_i^{\sigma_i} = (1-\sigma_i)\bar{z}_i + \sigma_i \bar{\epsilon}_i, \qquad \sigma_i \in [0,1], \tag{1}$$

$$\mathcal{L}_{\text{TF-AR}} = \mathbb{E}\Big[w(\sigma_i) \left\| v_\theta(x_i^{\sigma_i}, \sigma_i \mid h_{<i}, c) - (\bar{\epsilon}_i - z_i) \right\|_2^2 \Big]. \tag{2}$$

Here σ_i is the sampled noise level, v_θ the video velocity predictor, c the language condition, and w the scheduler weight. The recovery target retains the original z_i, teaching correction of rollout-like errors. Given one image and a language prompt, the model predicts coherent robot motion and object interactions (Appendix D), providing a predictive prior for action adaptation.

2.2 CAUSAL-TO-CAUSAL WORLD-ACTION ADAPTATION

Causal coupling. Following DreamZero (Ye et al., 2026b) and LingBot-VA (Li et al., 2026a), we couple video and action experts through an asymmetric interface. Video queries read only their own and earlier visual blocks, never action tokens; action queries read observed history, partially denoised futures, and the entire noisy action chunk. This preserves pretrained causal visual dependencies while grounding actions in both past and anticipated interaction. At a decision made at control step t, let Z_t^- denote observed video latents, q_t the robot state, c the language instruction, and $\mathbf{A}_t = a_{t:t+H-1}$ an H-step action chunk. The video expert θ predicts K_v future latent steps $\widetilde{Z}_t^+$ to noise level $\sigma_\star \in (0,1)$, then supplies their joint visual cache $\mathcal{K}_t$ to action expert ψ:

$$\begin{aligned} \widetilde{Z}_t^+ &= \text{Rollout}_{\theta,\sigma_\star}(\epsilon^v \mid Z_t^-, c, q_t), \quad \mathcal{K}_t = \text{Prefill}_\theta(Z_t^-, \widetilde{Z}_t^+; c, q_t, \sigma_\star), \\ \widehat{\mathbf{A}}_t &= \text{Denoise}_\psi(\epsilon^a \mid \mathcal{K}_t, c, q_t). \end{aligned} \tag{3}$$

Here ϵ^v and ϵ^a are independent Gaussian noise, $\mathcal{K}_t$ contains layer-wise video keys and values, and observed latents remain clean throughout. We refer to this predict-then-act mode as inverse dynamics modeling (IDM).

Training and inference. Training uses two passes: video flow matching conditioned on clean history, then action flow matching conditioned on history and a forward-noised ground-truth future, using $\sigma_\star = 0.9$. The second pass detaches the visual cache, so action loss updates only the action expert and proprioceptive adapter. Both branches predict noise-minus-data velocity, with weighted objective $\mathcal{L} = \lambda_v \mathcal{L}_{\text{video}} + \lambda_a \mathcal{L}_{\text{action}}$, where λ_v and λ_a balance the losses. At inference, the four video steps (V4) stop at $\sigma_\star = 0.9$ rather than at a clean video, and future latents are never decoded to pixels; their cache is reused throughout action denoising. Appendix C.2 specifies the training surrogate, masking, and cache scope.

2.3 CONTEXT SCALING OF WORLD-ACTION MODELS

We scale the duration of real observations in the causal prefix, keeping the visual forecast and action horizon fixed within each benchmark. Each window uses a separately trained model evaluated at its

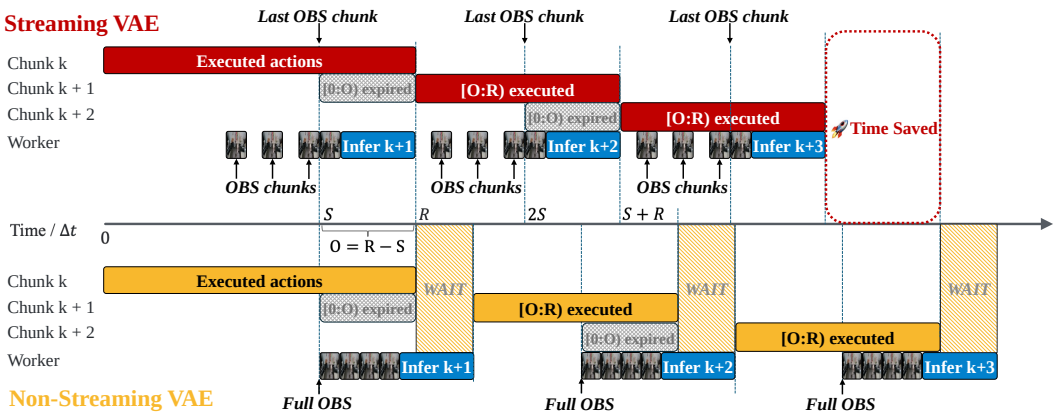

Figure 2: **Asynchronous model–robot execution with streaming VAE.** Both schedules use the same nominal trigger stride S and overlap $O = R - S$. Streaming VAE (top) encodes observation (OBS) chunks as they arrive and meets the handoff deadline $T_{\text{ready}} \leq O\Delta t$. Full-window encoding (bottom) delays handoffs and subsequent triggers, accumulating robot idle time. Time is shown in units of Δt.

training context length; zero history retains the current observation. Missing early-episode history repeats the initial frame.

We compare Wan2.2 (Team Wan et al., 2025) (bidirectional), LongLive-2.0 (AR), and LongLive2.0-Robot (robot-domain AR) initializations under causal WAM adaptation. Here, "bidirectional" describes the video model's pretraining, not the adapted policy's temporal mask. We study context scaling on LIBERO-Long (Liu et al., 2023) and RoboCasa GR-1 Tabletop (NVIDIA GEAR, 2025). RoboCasa GR-1 provides the most detailed curve, with tasks requiring successive object transfers and workspace changes. Longer prefixes increase prefill, attention, and cache costs; the following infrastructure section makes history-conditioned prediction practical for responsive execution.

3 Long-WAM Infrastructure: Real-Time Edge Deployment

Long-WAM combines asynchronous execution (Section 3.1) with shared optimizations and device-specific tuning (Section 3.2) for real-time control with visual history (Figures 2 and 3).

3.1 Asynchronous Execution

Pure asynchronous execution suffices. We overlap inference with robot execution without blending or prefix guidance. At decision k (control step t_k), the predicted chunk is $\widehat{\mathbf{A}}_{t_k} \in \mathbb{R}^{H \times d}$, where d is the action dimension (Section 2.2). Only the first R steps are eligible for execution; inference has nominal stride S ($R/2 \leq S < R \leq H$), leaving an overlap of $O = R - S$ steps. At the handoff, the controller discards the elapsed prefix and executes the new chunk's aligned suffix, waiting if it arrives late (Figure 2).

Pure asynchronous execution can disrupt action continuity (Motubrain Team, 2026), motivating inference-time guidance (Black et al., 2025b) and training-time action conditioning (Black et al., 2025c). Long-WAM requires neither in our experiments. Conditioned on temporally continuous LongLive2.0-Robot forecasts, consecutive chunks tend to agree in their overlap: on RoboTwin 2.0 (Chen et al., 2025a), Long-WAM nearly keeps its synchronous success under asynchrony, whereas Fast-WAM (Yuan et al., 2026b) and LingBot-VA (Li et al., 2026a) lose 15.4 and 45.5 percentage points (Table 6).

Streaming VAE. Reducing overlap from 12 to 8 control steps lowers overlap RMSE about fourfold and jerk nearly threefold (Appendix G). At fixed R, shorter overlap requires later inference triggers and a tighter handoff deadline:

$$T_{\text{ready}} \leq O\Delta t = (R - S)\Delta t, \tag{4}$$

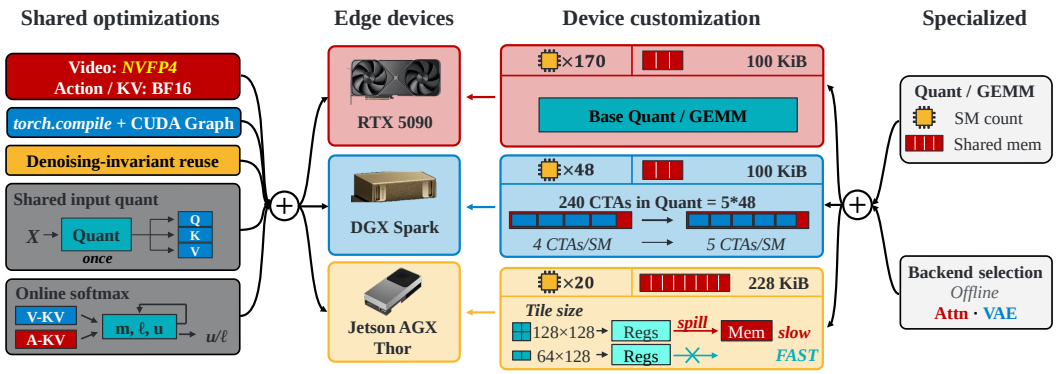

Figure 3: **Optimizations for efficient edge deployment.** Shared optimizations and device-specific tuning accelerate edge inference. "Base" denotes the Quant/GEMM implementation; shape-specific dispatch and autotuning remain enabled.

where Δt is the control interval and T_{ready} includes transfer, queueing, and computation from the time the trigger observation becomes available.

Streaming causal VAE encoding processes incoming frame groups. After the trigger, we encode the remaining frames, concatenate features, and project and normalize the latents. At the same S and O, it avoids full-window encoding's repeated waits and accumulated idle time (Figure 2), allowing later triggers and shorter overlap while retaining the latest observation.

3.2 Efficient Edge Deployment

Local inference avoids network delays, but video-first prediction is costly on limited onboard compute. We accelerate Long-WAM on NVIDIA GeForce RTX 5090, DGX Spark, and Jetson AGX Thor to support short-overlap execution (Equation 4) while preserving visual imagination. Shared optimizations reduce common computation and data-movement costs; device-specific tuning addresses platform constraints, balancing portability and hardware efficiency.

Shared optimizations. Across devices (Figure 3), video-expert linear layers use W4A4 NVFP4 (four-bit weights and activations) during generation and key–value (KV) prefill (NVIDIA, 2024; Chen et al., 2026b), while action compute and KV storage remain BF16. We combine PyTorch compilation (Ansel et al., 2024) and CUDA Graph replay (Gray, 2019) with denoising-invariant reuse of text/state KV, observed-video KV, and FP32 RoPE tables (Su et al., 2024) within each inference call to reduce execution overhead and redundant computation; streaming-VAE operations that update state remain outside CUDA Graph capture.

Shared input quantization quantizes the shared input to Q/K/V projections once and reuses the quantized activations and scales across three separate GEMMs. We combine attention over separate video and action KV buffers using online softmax (Milakov & Gimelshein, 2018) with a shared normalization, avoiding buffer concatenation. For each query, let (m_j, ℓ_j, u_j) denote the maximum attention logit, the sum of exponentials shifted by m_j, and their value-weighted sum for segment $j \in \{v, a\}$; the merged output is

$$m = \max(m_v, m_a), \qquad \text{Attn} = \frac{e^{m_v - m} u_v + e^{m_a - m} u_a}{e^{m_v - m} \ell_v + e^{m_a - m} \ell_a}.$$

This preserves joint attention without a concatenated KV buffer. We also coalesce memory accesses, fuse scale/bias and cast operations, and simplify VAE layouts and padding, subject to shape and backend constraints.

Device-specific tuning. Quant/GEMM tuning adjusts tile sizes, warps per block, pipeline stages, and buffers. Backend selection includes VAE layout and convolution tuning (Chetlur et al., 2014) and RTX 5090/Spark attention backends. RTX 5090 retains base Quant/GEMM implementations with shape-specific dispatch and execution tuning. Spark reduces quantization time with compact

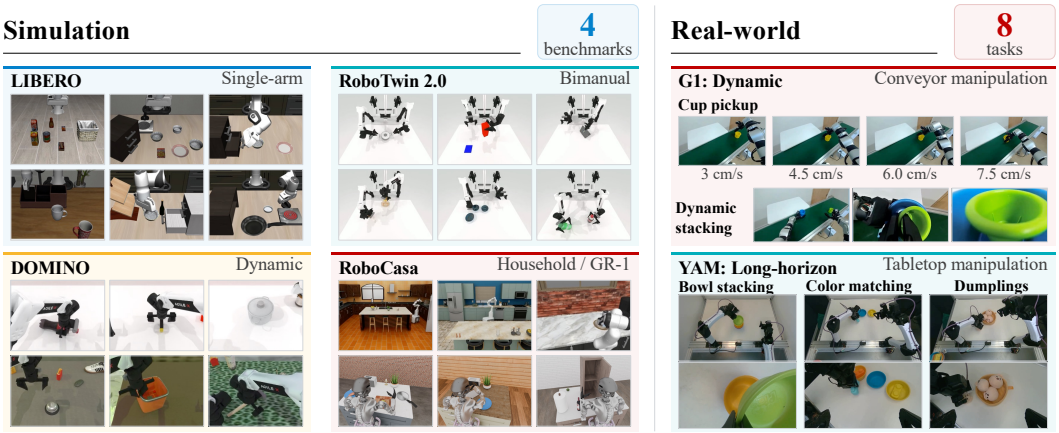

Figure 4: **Simulation environments and real-world tasks.** Our evaluation spans four simulation benchmarks—LIBERO, RoboTwin 2.0, DOMINO, and RoboCasa—and eight real-world task configurations on G1 and YAM. These include cup pickup at four conveyor speeds, dynamic cup stacking, bowl stacking, brick sorting by color, and dumpling placement into a pan.

Table 1: Success rate (SR, %) on **LIBERO**. w/o V: no future-video denoising; CoD: video–action co-denoising; IDM: inverse dynamics modeling.

Method	Spatial	Object	Goal	Long	Avg.
OpenVLA	84.7	88.4	79.2	53.7	76.5
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
GR00T-N1	94.4	97.6	93.0	90.6	93.9
π_0	96.8	98.8	95.8	85.2	94.1
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9
UniVLA	95.4	98.8	93.6	94.0	95.5
X-VLA	98.2	98.6	97.8	97.6	98.1
LingBot-VA	98.5	99.6	97.2	98.5	98.5
Motus	96.8	99.8	96.6	97.6	97.7
Fast-WAM	98.2	100.0	97.0	95.2	97.6
Long-WAM (w/o V)	98.0	99.5	97.0	94.5	97.3
Long-WAM (CoD)	98.6	99.8	96.8	95.8	97.8
Long-WAM (IDM)	**99.5**	**100.0**	**98.0**	**99.5**	**99.5**

Table 2: Success rate (SR, %) on **RoboTwin 2.0**.

Method	Clean	Rand.	Avg.
π_0	65.9	58.4	62.2
$\pi_{0.5}$	82.7	76.8	79.8
Motus	88.7	87.0	87.8
Motus (Wan2.2)	77.6	77.0	77.3
LingBot-VA	92.9	91.5	92.2
LingBot-VA 2.0	93.8	93.4	93.6
Fast-WAM	91.9	91.8	91.8
AHA-WAM	93.4	92.2	92.8
ABot-M0.5	94.0	94.2	94.1
Qwen-RobotManip	93.7	94.0	93.9
Long-WAM (w/o V)	92.4	91.6	92.0
Long-WAM (CoD)	94.0	93.1	93.6
Long-WAM (IDM)	**94.7**	**94.2**	**94.4**

buffers and per-shape choices of warps per block. More resident thread blocks per streaming multiprocessor (SM) allow the illustrated grid to run in one wave. Thor's larger per-block shared-memory budget supports GEMM configurations unavailable on Spark; smaller epilogue tiles avoid register spills. These adaptations reflect hardware constraints, not device-exclusive algorithms. Together, shared optimizations and device-specific tuning yield 3.2–4.1× total speedups over BF16 eager execution. Section 4.5 compares model latencies and cumulative gains (Tables 5 and 7).

4 Experiments

4.1 Implementation Details

Pretraining LongLive2.0-Robot uses approximately 30,720 GPU-hours on 64 NVIDIA H100 GPUs; downstream WAMs train on 16 GB200 GPUs. WAM training uses AdamW with a peak learning rate of 10^{-4}, cosine decay, 5% warmup, BF16 precision, and gradient clipping at 1.0. Robot experiments use a Unitree G1 humanoid and a YAM bimanual manipulator. Appendix E lists baseline references.

Figure 5: Context scaling with three video initializations. Context windows are displayed at equal spacing.

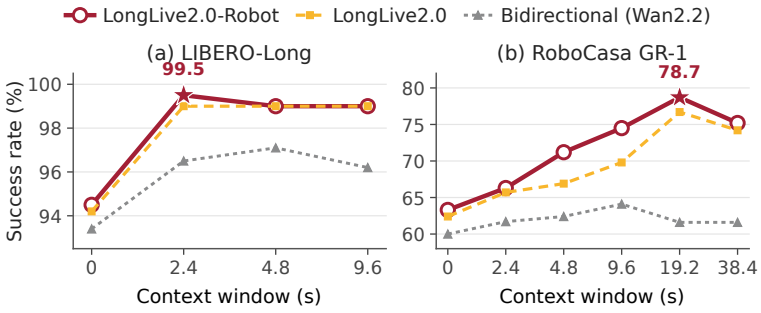

Table 4: Success rate (SR, %) on RoboCasa GR-1.

Method	SR ↑
π_0	62.5
Diffusion Policy	32.7
Fast-WAM	47.5
GR00T-N1.5	64.1
DreamZero	62.4
Cosmos Policy	67.1
Long-WAM (2.4 s)	**63.3**
Long-WAM (19.2 s)	**78.7**

4.2 Results on Simulation Benchmarks

We evaluate on LIBERO (Liu et al., 2023), RoboTwin 2.0 (Chen et al., 2025a), DOMINO (Fang et al., 2026), and RoboCasa GR-1 (NVIDIA GEAR, 2025) (Figure 4). For the first three benchmarks, the main results use up to 2.4 seconds of context; DOMINO adds moving objects, where recent motion informs anticipation and interception. RoboCasa GR-1, whose tasks chain several object transfers, hosts the longer-context study. We further evaluate compositional tasks on RoboCasa365, with GPT-6 Astra as a high-level planner (Appendix F, Table 9).

LIBERO. Long-WAM (IDM) achieves the highest average success (99.5%) and 99.5% on LIBERO-Long (Table 1). Its margin is largest on LIBERO-Long, the suite with the longest tasks, where it exceeds LingBot-VA (Li et al., 2026a) and Fast-WAM (Yuan et al., 2026b) by 1.0 and 4.3 points.

RoboTwin 2.0. Long-WAM (IDM) leads on average (94.4%) and Clean (94.7%), and ties the best Randomized score (94.2%; Table 2). Its average exceeds ABot-M0.5 and LingBot-VA 2.0 (Zhang et al., 2026a) by 0.3 and 0.8 points, respectively.

Table 3: DOMINO results after dynamic-data fine-tuning. Success rate (SR); manipulation score (MS).

Method	SR (%) ↑	MS ↑
OpenVLA	1.5	6.1
π_0	8.2	24.0
$\pi_{0.5}$	9.6	26.2
InternVLA-M1	5.4	27.6
OpenVLA-OFT	9.1	24.1
StarVLA-OFT	10.9	30.5
PUMA	17.2	35.0
Fast-WAM	19.9	33.3
Long-WAM	**34.9**	**45.1**

DOMINO. After dynamic-data fine-tuning, Long-WAM leads with 34.9% SR and 45.1 MS (Table 3), exceeding the strongest baseline on each metric: Fast-WAM by 15.0 SR points and PUMA (Fang et al., 2026) by 10.1 MS points. Recent observations supply motion cues that a single image cannot, supporting prediction-conditioned interception; Section 4.4 tests this ability on a physical conveyor.

4.3 Ablation Studies

Context Scaling. Adding 2.4 seconds of history raises LIBERO-Long success from 94.5% to 99.5% with LongLive2.0-Robot and from 94.2% to 99.0% with LongLive-2.0, after which the curve is flat (Figure 5). GR-1 keeps benefiting up to 19.2 seconds: success rises from 66.3% to 78.7% and from 65.7% to 76.7%, respectively, and the robot-video model exceeds the strongest GR-1 baseline by 11.6 points (Table 4). The useful window thus appears to follow the temporal structure of the task. At 38.4 seconds, success decreases to 75.2% and 74.2%; this window is three times the average training trajectory (12.1 seconds), and 80.4% of its sampled history frames are padding. We hypothesize that the decline reflects limited history coverage rather than an intrinsic memory limit. Context also has a price: 8× more history raises RTX 5090 latency 3.2× (107.4 to 341.0 ms; Appendix I).

Autoregressive vs. Bidirectional Pretraining. On GR-1, short context hides most of the difference between video foundations, and long context reveals it: the three initializations lie within 3.3

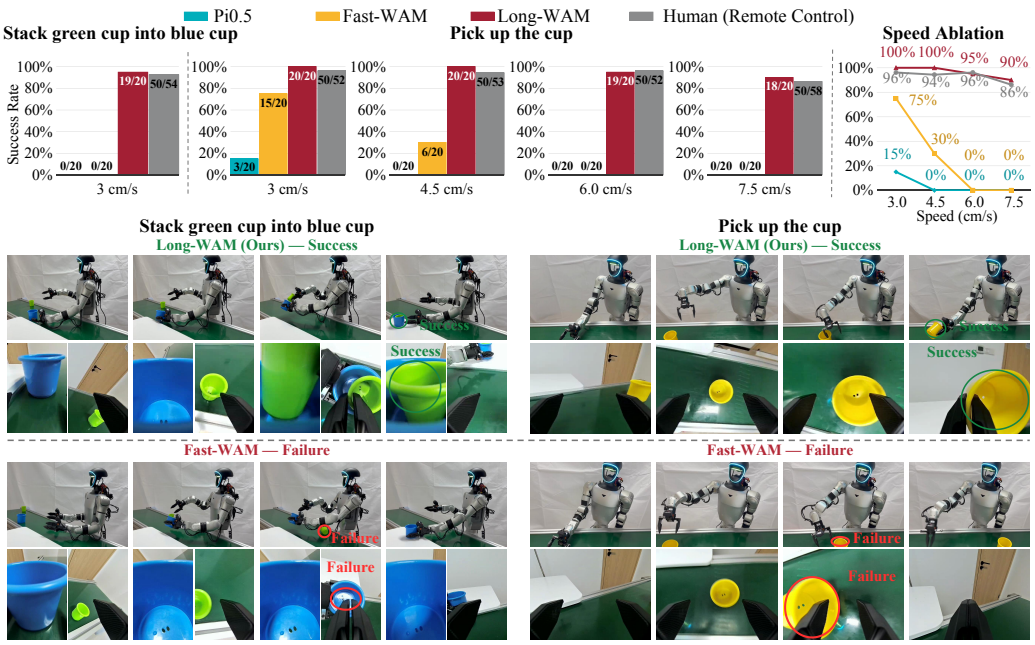

Figure 7: **Dynamic manipulation on Unitree G1.** Long-WAM maintains 90–100% grasping success across conveyor speeds and achieves 95% success on dynamic cup stacking. Top: policy and human teleoperation comparisons. Bottom: Long-WAM and Fast-WAM rollouts.

points without history (60.0–63.3%) but span 17.1 points at 19.2 seconds (61.6–78.7%; Figure 5). The bidirectional variant shows no net success-rate gain (61.7% at 2.4 seconds, 61.6% at 19.2, after a 64.1% peak at 9.6), whereas the AR variants gain 12.4 and 11.0 points. LIBERO-Long shows the same direction near its ceiling (gaps of 1.1 and 3.0 points at 0 and 2.4 seconds). Access to more history alone therefore does not produce these gains; the pattern is consistent with causal pretraining establishing history-to-future dependencies that downstream control can exploit.

Video–Action Denoising Strategy. IDM's advantage concentrates on the longest tasks. On the three shorter LIBERO suites, the three variants lie within 1.5 points; on LIBERO-Long, IDM reaches 99.5%, versus 97.8% for co-denoising (CoD) and 94.5% without future video (w/o V; Tables 1 and 2). IDM costs 16.5 ms more than CoD per chunk on RTX 5090 (Appendix H), a cost justified by its higher success on LIBERO-Long (Table 1).

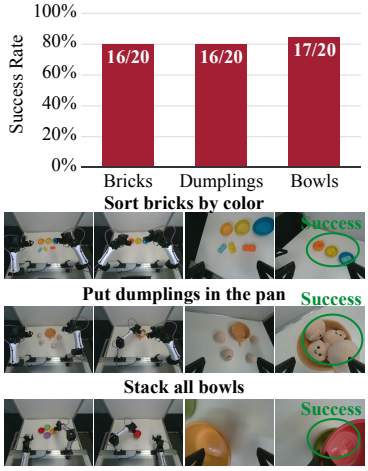

Figure 6: **Long-horizon execution on YAM:** 20 trials per task.

4.4 Real-world Deployment

We evaluate dynamic manipulation on Unitree G1 and long-horizon tasks on YAM, with 20 trials per policy and condition.

Dynamic Tasks. Conveyor speed separates the policies (Figure 7). As the belt accelerates from 3.0 to 7.5 cm/s, the grasping success of Fast-WAM (Yuan et al., 2026b) falls from 75% to 0% and that of $\pi_{0.5}$ (Physical Intelligence et al., 2025) from 15% to 0%, whereas Long-WAM stays at 90–100% (100%, 100%, 95%, and 90%). Stacking a moving green cup into a blue cup at 3 cm/s adds alignment and placement to interception; Long-WAM succeeds in 19 of 20 trials, and neither baseline succeeds once.

Table 5: RTX 5090 end-to-end latency and RoboTwin 2.0 success rate (SR; Clean/Randomized mean). BF16 eager is the unoptimized V4/A4 baseline.

Method	Latency (ms) ↓	SR (%) ↑
LingBot-VA	3618.4	92.2
Motus	1201.1	87.8
Cosmos Policy	470.6	–
Fast-WAM	244.1	91.8
Long-WAM		
BF16 eager	356.0	94.4
Optimized (V4/A4)	**107.4**	**93.5**
Optimized (V2/A2)	**81.8**	**92.5**

Table 6: RoboTwin 2.0 success rate (SR, %). Sync: Table 2; Async: our evaluation. Jerk denotes a discrete smoothness proxy; see Appendix G.

Method	Sync	Async		
	SR ↑	SR ↑	RMSE ↓	Jerk ↓
Fast-WAM	91.8	76.4	0.0731	0.1316
LingBot-VA	92.2	46.7	0.1432	0.6592
Long-WAM (CoD)	93.6	93.2	0.0260	0.0479
Long-WAM (IDM)	**94.4**	**94.2**	**0.0246**	**0.0436**

Long-Horizon Tasks. On YAM, where tasks last over 40 seconds on average, Long-WAM succeeds in 80%, 80%, and 85% of trials on brick sorting by color, placing dumplings in a pan, and stacking bowls (81.7% on average; Figure 6), sustaining multi-step execution and rapid interception.

4.5 Efficiency

Table 7: End-to-end latency (E2E) of Long-WAM under cumulative optimizations across devices, including the full VAE computation. Each row keeps all preceding optimizations on a fixed input; speedups are relative to BF16 eager.

Optimization	RTX 5090		DGX Spark		Thor	
	E2E (ms) ↓	Speedup ↑	E2E (ms) ↓	Speedup ↑	E2E (ms) ↓	Speedup ↑
Shared optimizations						
BF16 eager	356.0	1.0×	1342.8	1.0×	1215.2	1.0×
+ NVFP4 quantization	360.4	1.0×	1147.4	1.2×	1219.8	1.0×
+ CUDA Graph + compile	172.4	2.1×	686.4	2.0×	762.4	1.6×
+ Denoising-invariant reuse	144.9	2.5×	464.5	2.9×	524.5	2.3×
+ Shared input quantization	130.3	2.7×	427.8	3.1×	476.5	2.6×
+ Video–action online softmax	**126.9**	**2.8**×	**419.9**	**3.2**×	**468.2**	**2.6**×
Device-specific tuning						
+ Quant/GEMM tuning	119.1	3.0×	415.6	3.2×	458.9	2.6×
+ Backend selection	**107.4**	**3.3**×	**328.2**	**4.1**×	**378.7**	**3.2**×

Latency. On RTX 5090, the optimizations detailed in Section 3.2 jointly reduce the V4/A4 end-to-end inference latency to 107.4 ms while largely preserving SR (BF16: 94.4%; optimized: 93.5%; Table 5). Long-WAM achieves a 2.3× speedup over Fast-WAM (Yuan et al., 2026b) (244.1 ms) despite additionally predicting future video frames. Both optimized variants outperform existing baselines in both inference latency and reported SR; notably, V2/A2 trades a modest 1.0-point drop in SR for an additional 24% latency reduction (81.8 ms). The other baselines are evaluated under their native deployment configurations and denoising budgets.

Pure Asynchronous Execution. Long-WAM largely preserves synchronous success. At $S = 12$ ($R = 24$), IDM loses 0.2 percentage points (94.4% to 94.2%) and CoD loses 0.4; Fast-WAM and LingBot-VA (Li et al., 2026a) lose 15.4 and 45.5 (Table 6). Long-WAM's overlap RMSE and jerk are about one-third of Fast-WAM's. With a shorter overlap ($S = 16$), asynchronous success reaches 95.0%, on par with synchronous execution (Appendix G).

Cumulative Acceleration. NVFP4 alone does not reduce latency on RTX 5090 or Thor. Shared optimizations reduce latency to 126.9, 419.9, and 468.2 ms on RTX 5090, Spark, and Thor (Table 7). Device-specific tuning reduces latency by another 15–22%, yielding 107.4, 328.2, and 378.7 ms, for total speedups of 3.3×, 4.1×, and 3.2× over BF16 eager. Appendix H details timing and configuration-dependent KV reuse effects.

AI USE STATEMENT

In this work, We used generative AI tools for language polishing, including improving grammar, clarity, and readability. All research ideas, theoretical developments, mathematical claims and proofs, method implementations, experimental design and execution, and analyses of results were carried out by the authors without generative AI assistance. All AI-assisted edits were reviewed and revised as needed by the authors. We take full responsibility for the final content of this paper, including its claims, results, and conclusions.

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. AgiBot World Colosseo: A Large-scale Manipulation Platform for Scalable and Intelligent Embodied Systems, 2025. URL https://arxiv.org/abs/2503.06669.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947. ACM, 2024. doi: 10.1145/3620665.3640366. URL https://doi.org/10.1145/3620665.3640366.

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, 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 Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems XXI*, RSS2025. Robotics: Science and Systems Foundation, June 2025a. doi: 10.15607/rss.2025.xxi.010. URL http://dx.doi.org/10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems 38*, NeurIPS 2025, pp. 37596–37620. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025b. doi: 10.52202/085713-1122. URL http://dx.doi.org/10.52202/085713-1122.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-Time Action Conditioning for Efficient Real-Time Chunking, 2025c. URL https://arxiv.org/abs/2512.05964.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. AHA-WAM:Asynchronous Horizon-Adaptive World-Action Modeling with Observation-Guided Context Routing, 2026. URL https://arxiv.org/abs/2606.09811.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, Tianlun Li, Mingxin Wang, Dekang Qi, Bin Hu, Wei Mei, Yuze Xuan, Haolong Yang, Yanqing Zhu, Mu Xu, Zhiheng Ma, and Xinyuan Chang. ABot-M0.5: Unified Mobility-and-Manipulation World Action Model, 2026a. URL https://arxiv.org/abs/2607.00678.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, 2025a. URL https://arxiv.org/abs/2506.18088.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, Bolun Wang, Fangjing Wang, Hanqing Wang, Tai Wang, Ziqin Wang, Xueyuan Wei, Chao Wu, Shuai Yang, Jinhui Ye, Junqiu Yu, Jia Zeng, Jingjing Zhang, Jinyu Zhang, Shi Zhang, Feng Zheng, Bowen Zhou, and Yangkun Zhu. InternVLA-M1: A Spatially Guided Vision-Language-Action Framework for Generalist Robot Policy, 2025b. URL https://arxiv.org/abs/2510.13778.

Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, Yuyang Zhao, Huizi Mao, Ying-Cong Chen, Enze Xie, Xiaojuan Qi, and Song Han. LongLive-2.0: An NVFP4 Parallel Infrastructure for Long Video Generation, 2026b. URL https://arxiv.org/abs/2605.18739.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient Primitives for Deep Learning, 2014. URL https://arxiv.org/abs/1410.0759.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems XIX*, RSS2023. Robotics: Science and Systems Foundation, July 2023. doi: 10.15607/rss.2023.xix.026. URL http://dx.doi.org/10.15607/RSS.2023.XIX.026.

Yufan Deng, Zilin Pan, Hongyu Zhang, Xiaojie Li, Ruoqing Hu, Yufei Ding, Yiming Zou, Yan Zeng, and Daquan Zhou. Rethinking Video Generation Model for the Embodied World, 2026. URL https://arxiv.org/abs/2601.15282.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards Generalizable Robotic Manipulation in Dynamic Environments. In *European Conference on Computer Vision (ECCV)*, 2026.

Yusen Feng, Bingchen Han, Jiangran Lyu, Kai Liu, Yixin Zheng, Yuxuan Wan, Weiheng Liu, Sun Han, Ruiqin Li, Yulong Zhang, Fangfu Liu, Xuesong Shi, Libin Liu, Yizhou Wang, Zhizheng Zhang, and He Wang. WAM-TTT: Steering World-Action Models by Watching Human Play at Test Time, 2026. URL https://arxiv.org/abs/2607.06988.

Alan Gray. Getting Started with CUDA Graphs. NVIDIA Technical Blog, 2019. URL https://developer.nvidia.com/blog/cuda-graphs/.

Ryan Hoque, Peide Huang, David Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning Dexterous Manipulation from Large-Scale Egocentric Video. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 4218–4237, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/07fcc6e2b89439d3ee5ab60939aaa6a0-Paper-Conference.pdf.

Muyan Hu, Ashwin Venkatram, Shreyashri Biswas, Balamurugan Marimuthu, Bohan Hou, Gabriele Oliaro, Haojie Wang, Liyan Zheng, Xupeng Miao, Jidong Zhai, and Zhihao Jia. Optimal Kernel Orchestration for Tensor Programs with Korch. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 755–769. ACM, 2024. doi: 10.1145/3620666.3651383. URL https://doi.org/10.1145/3620666.3651383.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, and Linxi "Jim" Fan. RoboTTT: Context Scaling for Robot Policies, 2026. URL https://arxiv.org/abs/2607.15275.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, Donguk Lee, Heeseung Kwon, Hojin Jeon, Jaehyun Kang, Jaekyoung Bae, Jihyuk Lee, Jimin Lee, John Won, Joonwoo Ahn, Junhyeong Park, Junyoung Sung, Kyungmin Lee, Minseong Han, Minsung Yoon, Sejune Joo, Seonil Son, Seungcheol Park, Seunggeun Cho, Seungjun Moon, Seungku Kim, Yonghoon Dong, Yongjin Cho, Youngchan Kim, Chang Hwan Kim, Dohyeon Kim, Heecheol Kim, Heewon Lee, Hensen Ahn, Hyungkyu Ryu, Hyunsoo Choi, Hyunsoo Shin, Jaeheon Jung, Jaewoo Kim, Jinwook Kim, Joochul Chang, Joonsoo Kim, Junghun Park, Jungwoo Park, Junho Cho, Junhyeok Park, Junwon Lee, Kangwook Lee, Kwanghoon Kim, Kyoungwhan Choe, Manoj Bhadu, Nayoung Oh, Sangjun Kim, Sangwoo Kim, Seunghoon Shim, Seunghyun Kim, Seungjun Lee, Seungyup Ka, Sungryol Yang, Wook Jung, Yashu Shukla, Yeonjae Lee, Yeonwoo Bae, and Jinwoo Shin. RLDX-1 Technical Report, 2026a. URL https://arxiv.org/abs/2605.03269.

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, 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, 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, 2026b. URL https://arxiv.org/abs/2601.16163.

Wayne King, Zeyue Xue, Yuxuan Bian, Jie Huang, Haoran Li, Yaowei Li, Yaofeng Su, Yuming Li, Haoyu Wang, Shiyi Zhang, Songchun Zhang, Yuwei Niu, Sihan Xu, Junhao Zhuang, Haoyang Huang, and Nan Duan. Echo-Memory: A Controlled Study of Memory in Action World Models, 2026. URL https://arxiv.org/abs/2606.09803.

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, 2026a. URL https://arxiv.org/abs/2601.21998.

Qixiu Li, Yu Deng, Yaobo Liang, Lin Luo, Lei Zhou, Chengtang Yao, Lingqi Zeng, Zhiyuan Feng, Huizhi Liang, Sicheng Xu, Yizhong Zhang, Xi Chen, Hao Chen, Lily Sun, Dong Chen, Jiaolong Yang, and Baining Guo. Scalable Vision-Language-Action Model Pretraining for Robotic Manipulation with Real-Life Human Activity Videos, 2025. URL https://arxiv.org/abs/2510.21571.

Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable Video Infinity: Infinite-Length Video Generation with Error Recycling. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 23406–23432, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2858f8c8683aaa8c12d487354cf328dc-Paper-Conference.pdf.

Xuewu Lin, Tianwei Lin, Yun Du, Hongyu Xie, Yiwei Jin, Jiawei Li, Shijie Wu, Qingze Wang, Mengdi Li, Mengao Zhao, Ziang Li, Chaodong Huang, Hongzhe Bi, Lichao Huang, and Zhizhong Su. HoloBrain-0 Technical Report, 2026. URL https://arxiv.org/abs/2602.12062.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/

paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

Motubrain Team. World Action Models in Real Time: An Empirical Study of Smooth Execution via Asynchronous Deployment, 2026. URL https://arxiv.org/abs/2608.01880.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.050. URL https://www.roboticsproceedings.org/rss20/p050.html.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a05003fdb1e9562ab0c0a9719ea4de10-Paper-Conference.pdf.

NVIDIA. NVIDIA Blackwell Architecture Technical Brief, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture/blackwell-architecture-technical-brief.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots, 2025. URL https://arxiv.org/abs/2503.14734.

NVIDIA GEAR. PhysicalAI-Robotics-GR00T-Teleop-Sim: Simulation GR1 Tabletop Task 1K Dataset. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/nvidia/PhysicalAI-Robotics-GR00T-Teleop-Sim.

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

Ryan Punamiya, Simar Kareer, Zeyi Liu, Josh Citron, Ri-Zhao Qiu, Xiongyi Cai, Alexey Gavryushin, Jiaqi Chen, Davide Liconti, Lawrence Y. Zhu, Patcharapong Aphiwetsa, Baoyu Li, Aniketh Cheluva, Pranav Kuppili, Yangcen Liu, Dhruv Patel, Aidan Gao, Hye-Young Chung, Ryan Co, Renee Zbizika, Jeff Liu, Xiaomeng Xu, Haoyu Xiong, Geng Chen, Sebastiano Oliani, Wenkai Xuan, Chenyu Yang, Xi Wang, James Fort, Richard Newcombe, Josh Gao, Jason Chong, Garrett Matsuda, Aseem Doriwala, Marc Pollefeys, Robert Katzschmann, Xiaolong Wang, Shuran Song, Judy Hoffman, and Danfei Xu. EgoVerse: An Egocentric Human Dataset for Robot Learning from Around the World, 2026. URL https://arxiv.org/abs/2604.07607.

RoboCasa Team. RoboCasa365 Leaderboard. https://robocasa.ai/leaderboard.html, 2026. Snapshot updated September 12, 2026; accessed September 16, 2026.

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing, 2026. URL https://arxiv.org/abs/2604.05014.

Haisheng Su, Zongdai Liu, Xin Jin, Haoxuan Dou, Chengming Hu, Baorun Li, Zhanwang Liu, Ruiyan Xu, Jianjie Fang, Xin Zhang, Zhenjie Yang, Xue Yang, Chen Gao, Junchi Yan, Yong Li, and Wei Wu. WorldScape Policy 2.0: Empowering Steerable World Action Modeling with Reasoning-Augmented Memory, 2026. URL https://arxiv.org/abs/2607.18840.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen. HiMem-WAM: Hierarchical Memory-Gated World Action Models for Robotic Manipulation, 2026a. URL https://arxiv.org/abs/2606.10363.

Zhenguo Sun, Yu Sun, Hande Huang, and Alois Knoll. ω-EVA: Envision, Verify, and Act with Latent Interactive World Models, 2026b. URL https://arxiv.org/abs/2606.09457.

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

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Jiabing Yang, Zhaowen Li, Yan Huang, Liang Wang, and Peng Su. DIM-WAM: World-Action Modeling with Diverse Historical Event Memory, 2026a. URL https://arxiv.org/abs/2606.27677.

Ruixiang Wang, Qingming Liu, Yueci Deng, Guiliang Liu, Zhen Liu, and Kui Jia. EVA: Aligning Video World Models with Executable Robot Actions via Inverse Dynamics Rewards, 2026b. URL https://arxiv.org/abs/2603.17808.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified Vision-Language-Action Model, 2025. URL https://arxiv.org/abs/2506.19850.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient World Action Modeling with Persistent Memory, 2026. URL https://arxiv.org/abs/2606.20562.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An Efficient Action-Centered World–Action Model, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, 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, 2026b. URL https://arxiv.org/abs/2602.15922.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu

Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-RobotManip Technical Report: Alignment Unlocks Scale for Robotic Manipulation Foundation Models, 2026a. URL https://arxiv.org/abs/2606.17846.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination?, 2026b. URL https://arxiv.org/abs/2603.16666.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, Jiaming Zhou, Yishu Shen, Yudong Jin, Fangyi Xu, Shuailei Ma, Jiaqi Liao, Guanxing Lu, Zifan Shi, Yongkun Wen, Yujie Zhao, Weixuan Tang, Xinyang Wang, Chaojian Li, Jiapeng Zhu, Ka Leong Cheng, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Native Video-Action Pretraining for Generalizable Robot Control, 2026a. URL https://arxiv.org/abs/2607.08639.

Yang Zhang, Jiangyuan Zhao, Chenyou Fan, Fangzheng Yan, Tian Li, Haitong Tang, Sen Fu, Xuan'er Wu, Qizhen Weng, Weinan Zhang, Xiu Li, Chi Zhang, Chenjia Bai, and Xuelong Li. PRTS: A Primitive Reasoning and Tasking System via Contrastive Representations, 2026b. URL https://arxiv.org/abs/2604.27472.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Rui Wan, Widyadewi Soedarmadji, Enshu Liu, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 65811–65841, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a4a1ee071ce0fe63b83bce507c9dc4d7-Paper-Conference.pdf.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-Prompted Transformer as Scalable Cross-Embodiment Vision-Language-Action Model, 2025. URL https://arxiv.org/abs/2510.10274.

Deyu Zhou, Quan Sun, Yuang Peng, Kun Yan, Runpei Dong, Duomin Wang, Zheng Ge, Nan Duan, and Xiangyu Zhang. Taming Teacher Forcing for Masked Autoregressive Video Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7374–7384, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhou_Taming_Teacher_Forcing_for_Masked_Autoregressive_Video_Generation_CVPR_2025_paper.html.

A Discussion and Conclusion

Long-WAM studies context scaling where memory must ultimately serve action. The empirical gains on LIBERO-Long, RoboCasa GR-1, and real-world manipulation make recent physical history a useful resource for world-action modeling, while the deployment system connects that resource to responsive execution. This shifts the design question from whether a policy has memory to how its context, predictive representation, and execution budget should be designed together.

The goal is not unlimited history in the low-level controller. A manipulation policy benefits from remembering motion and recent interaction state, whereas hour-scale goals, reasoning, and task decomposition are naturally handled by a higher-level planner. These roles are complementary: a planner can maintain semantic continuity while Long-WAM executes local objectives with the physical context needed for control. The appropriate window may therefore depend on the task and the available compute, rather than follow a universal duration.

Each context window in our study is trained as its own model, which gives a clean view of how much history helps at every length; the natural next step is a single policy that adjusts its window at test time. The measured latency profile (74.6 ms without history, 341.0 ms at 19.2 seconds; Appendix I) indicates how much computation such adaptivity could reclaim. The decline at 38.4 seconds coincides with sparse real history in current datasets (80.4% padding), so longer and denser robot recordings offer a direct route to extending the useful window. With the streaming deployment stack in place, real-robot context-length studies are a natural next step, showing how the simulated trends carry over to physical interaction.

The observed trends motivate adaptive context allocation: preserving the history relevant to the current interaction while matching computation to its response requirements. They do not prescribe a power law or a fixed memory optimum. More broadly, Long-WAM motivates evaluating generative robot policies jointly by what they remember, how well they act, and how quickly they can incorporate new observations.

B Related Work

World-action modeling from video generators. WAMs connect visual dynamics to executable actions. Motus (Bi et al., 2025) combines pretrained video, vision–language, and action experts; DreamZero (Ye et al., 2026b) and LingBot-VA (Li et al., 2026a) adapt bidirectional video backbones to causal control. Their video pretraining supplies visual priors, while causal prediction and action coupling are learned downstream. LingBot-VA 2.0 (Zhang et al., 2026a) instead jointly pretrains causal video and learned latent actions in a shared representation. EVA (Wang et al., 2026b) aligns video generation with executable actions through inverse-dynamics rewards, and ω-EVA (Sun et al., 2026b) evaluates latent action consequences. Long-WAM develops a staged route: action-free robot-video AR pretraining followed by action adaptation under the same causal ordering. The action model thus inherits a visual dynamics prior explicitly trained for history-to-future prediction, separating predictive pretraining from learning an embodiment's action representation.

Memory and context scaling. Cache capacity, physical history duration, and the cost of rebuilding visual context are distinct quantities. Causal caches, compressed histories, and event retrieval provide complementary memory interfaces (Ye et al., 2026b; Li et al., 2026a; Yang et al., 2026; Sun et al., 2026a; Wang et al., 2026a; Su et al., 2026). Echo-Memory (King et al., 2026) studies memory for camera-conditioned video generation; RoboTTT (Jiang et al., 2026) studies context scaling in robot policies; WAM-TTT (Feng et al., 2026) adapts memory from human video to steer a frozen WAM. Long-WAM studies the robot's own observed interaction history: we vary its temporal extent across trained causal WAM variants and examine closed-loop success and online computation. Our focus is how predictive pretraining shapes the benefit of longer context, complementing work on memory compression, retrieval, and adaptation.

Efficient and deployable generative control. Large video backbones make online WAM control expensive. Prior systems reduce this cost through caching, few-step generation, pipelined execution, action-only decoding, or hierarchical update rates (Ye et al., 2026b; Li et al., 2026a; Yuan et al., 2026b; Cai et al., 2026). LongLive-2.0 further develops parallel AR training, compressed KV

caches, low-precision execution, and asynchronous VAE decoding (Chen et al., 2026b). Complementary work on compilation, kernel orchestration, and video-DiT quantization reduces inference cost (Ansel et al., 2024; Hu et al., 2024; Zhao et al., 2025). Asynchronous deployment additionally requires consistency between consecutive action chunks. Existing methods constrain these transitions through inference-time guidance, prefix-conditioned training, or denoising-time blending (Black et al., 2025b;c; Lin et al., 2026). An empirical WAM study highlights the importance of temporal alignment and the precision–smoothness trade-offs of transition strategies (Motubrain Team, 2026). Long-WAM exhibits robust continuity under direct asynchronous switching while retaining explicit future visual conditioning. We preserve this imagine-then-act path and address its sequential cost through streaming observation encoding and device-specific acceleration, targeting continuity and speed together.

C Data and Model Training Details

C.1 Pretraining Data

The pretraining corpus contains 2,294,889 model-ready text-and-image-to-video samples from five dataset families. Table 8 records the six source subsets, including two AgiBot World subsets. The reported training, validation, and test splits contain 2,251,021, 21,931, and 21,937 samples, respectively. The approximately 10,000-hour training scale is a window-equivalent estimate: assigning 16 seconds to each training sample yields 10,004.5 aggregate hours. It is not a measurement of unique raw footage; overlapping windows, source clip lengths, and padding affect that distinction.

Table 8: Pretraining data composition. Counts describe the full corpus before the train/validation/test split, not the published size of each source dataset.

Source subset	Samples
RoVid-X (Deng et al., 2026)	1,898,562
AgiBot World: `full146` (AgiBot-World-Contributors et al., 2025)	69,834
AgiBot World: 5.2–32 s supplement	40,497
EgoDex (Hoque et al., 2026)	207,044
EgoVerse (Punamiya et al., 2026)	59,742
VITRA (Li et al., 2025)	19,210
Total	2,294,889

Pretraining uses video prediction without action supervision, so sources need not share action coordinates or actuator dimensions. Action-space-independent supervision is a property of this stage; improved transfer to an unseen embodiment would require a separate evaluation.

C.2 Training Objectives and Inference Interface

Video pretraining. Let z_i denote an uncorrupted target chunk and $\epsilon_i \sim \mathcal{N}(0, I)$ its base Gaussian noise. With context, latent, and noise perturbations δ_i^c, δ_i^z, and δ_i^ϵ, the error-recycling inputs are

$$\bar{\epsilon}_i = \epsilon_i + \delta_i^\epsilon, \qquad x_i^{\sigma_i} = (1-\sigma_i)(z_i + \delta_i^z) + \sigma_i \bar{\epsilon}_i, \qquad h_{<i} = (z_j + \delta_j^c)_{j<i}. \tag{5}$$

Perturbations are sampled from buffered model errors when their corresponding augmentation is enabled, and are zero otherwise. The velocity target is $\bar{\epsilon}_i - z_i$, not $\bar{\epsilon}_i - (z_i + \delta_i^z)$: latent-input corruption is corrected toward the original target. The conditioning image remains unperturbed and is excluded from the loss. Paired clean/noisy streams implement teacher forcing: a target block accesses preceding context blocks and its own noisy tokens, without accessing later blocks or its own clean target. Loss is averaged over eligible target elements and weighted by the scheduler, as in Eq. 2.

World-action adaptation. Both branches use the flow path $x^\sigma = (1-\sigma)x + \sigma\epsilon$ and target $u = \epsilon - x$. For branch $b \in \{v, a\}$, let m_b select valid supervised elements and r_b be its prediction residual. The masked loss is

$$\mathcal{L}_b = \mathbb{E}\left[w_b(\sigma_b) \frac{\|m_b \odot r_b\|_2^2}{\max(\|m_b\|_1, 1)} \right], \qquad r_b = v_b(x_b^{\sigma_b}, \sigma_b \mid \text{conditioning}) - (\epsilon_b - x_b). \tag{6}$$

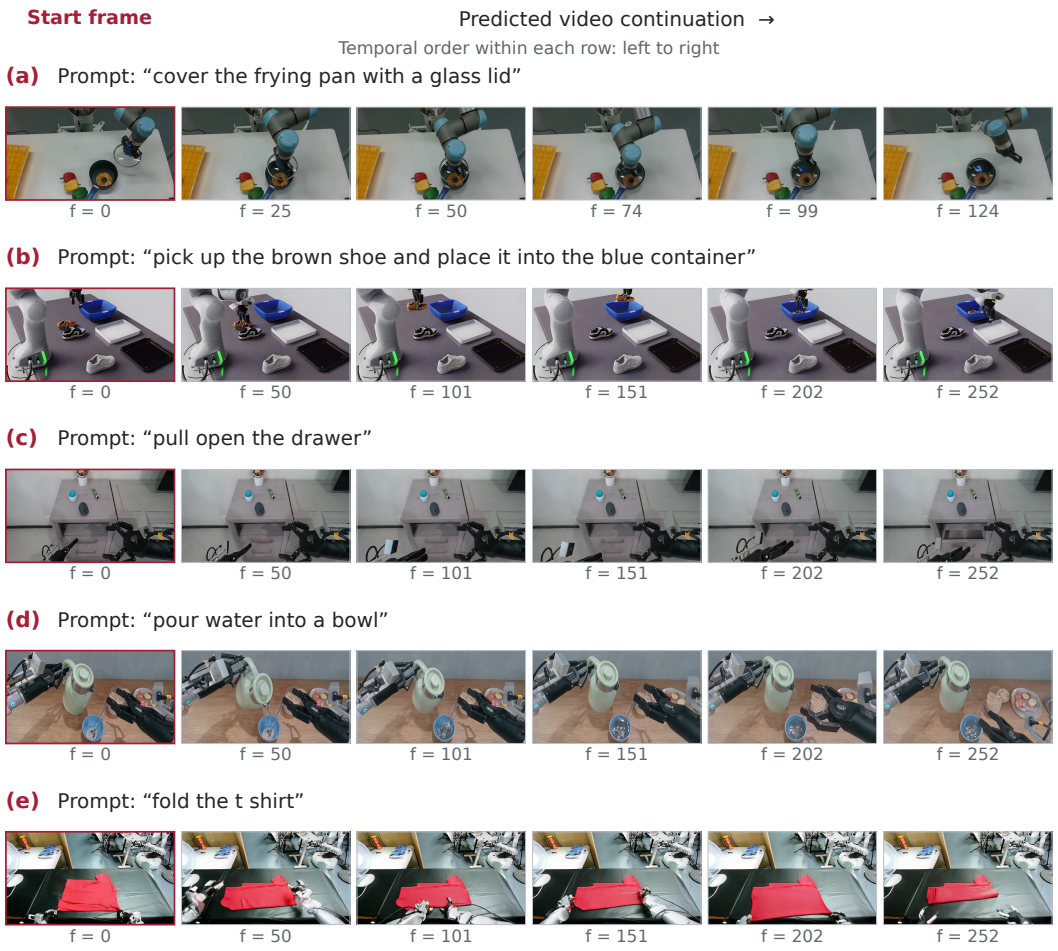

Figure 8: **Task-conditioned video prediction with LongLive2.0-Robot.** Five selected text-and-image-to-video (TI2V) examples, with the task prompt shown above each row. f denotes the zero-based frame index. All frames retain the original field of view.

Here $\odot$ is element-wise multiplication, and the temporal validity mask is broadcast over feature dimensions. It excludes fully padded future latent steps and padded action timesteps. The video pass supervises noisy future latents with the observed prefix clamped clean. The action pass uses a detached video cache built from the observed prefix and ground-truth future latents forward-noised to $\sigma_\star = 0.9$. It updates the action expert and the shared proprioceptive adapter through the action-conditioning path, without backpropagating into the video cache. The two-pass objective does not use the paired-stream or error-recycling augmentations of video pretraining.

At inference, partial video rollout starts from Gaussian noise and stops at $\sigma_\star$; it does not observe ground-truth future frames. Training and inference share this noise level but not the source of the future latents: the training surrogate retains a residual ground-truth component. This is a training approximation, not an assertion of identical conditioning distributions. Future latents remain in latent space, and the resulting visual KV cache is reused throughout joint action-chunk denoising.

Cache scope. The implementation appends a projection of the latest robot state q_t to the conditioning tokens of both experts. Consequently, visual features can depend on q_t at multiple layers. Reuse within one action solve is distinct from reuse across decisions: updating q_t, evicting old context, or changing temporal positions can invalidate previously computed visual keys and values. Any persistent-cache optimization must specify its refresh semantics and maintain the intended observation/position alignment.

D LONGLIVE2.0-ROBOT VIDEO PREDICTIONS

Table 9: **Success rate (SR, %) on RoboCasa365.** Overall averages 50 tasks: 18 Atomic-Seen, 16 Composite-Seen, and 16 Composite-Unseen. Baseline sources: Appendix E.

Method	Atomic Seen	Composite Seen	Composite Unseen	Overall
Diffusion Policy	15.7	0.2	1.3	6.1
Azero-Robotics-1	30.3	3.8	1.6	12.6
π_0	34.6	6.1	1.1	14.8
GigaWorld-Policy 0.1	44.4	11.8	2.9	20.7
GR00T N1.6	51.1	9.4	1.7	21.9
GR00T N1.5	50.7	14.8	2.7	23.9
WorldDreamer	66.3	26.7	9.0	35.3
RLDX-1	67.6	27.9	8.5	36.0
PRTS	66.3	30.3	18.8	39.6
Qwen-RobotManip	68.6	20.1	14.9	35.9
ABot-M0.5	75.6	37.7	3.3	40.3
GPT-6 Astra	31.5	22.4	20.8	25.2
$\pi_{0.5}$	39.6	7.1	1.2	16.9
$\pi_{0.5}$ + GPT-6 Astra	46.5	24.0	19.0	30.5
Long-WAM	67.9	15.8	6.1	31.4
Long-WAM + GPT-6 Astra	**85.6**	**38.8**	**35.0**	**54.4**

Given a single image and a language instruction, LongLive2.0-Robot predicts how a robot interacts with its environment (Figure 8). The selected rollouts follow the prompted tasks: the gripper positions a lid over a pan and withdraws, transfers the specified shoe into a container, and pulls a drawer open. Pouring and T-shirt folding further illustrate coordinated motion involving changing object configurations. Across these examples, the predicted arm movements and object transitions form coherent, task-directed sequences rather than merely preserving scene appearance. These qualitative results provide evidence that robot-video pretraining learns a predictive representation of robot motion and object interaction, supplying a task-conditioned visual prior for subsequent world-action adaptation.

E BASELINE DETAILS

Our policy baselines include Diffusion Policy (Chi et al., 2023), OpenVLA (Kim et al., 2024), OpenVLA-OFT (Kim et al., 2025), GR00T (NVIDIA et al., 2025), π_0 (Black et al., 2025a), $\pi_{0.5}$ (Physical Intelligence et al., 2025), UniVLA (Wang et al., 2025), X-VLA (Zheng et al., 2025), InternVLA-M1 (Chen et al., 2025b), and StarVLA (StarVLA Community, 2026). UniVLA refers to Wang et al.'s *Unified Vision-Language-Action Model*; PUMA is the method introduced with DOMINO (Fang et al., 2026).

World-action baselines include Motus (Bi et al., 2025), LingBot-VA (Li et al., 2026a), LingBot-VA 2.0 (Zhang et al., 2026a), Fast-WAM (Yuan et al., 2026b), AHA-WAM (Cai et al., 2026), DreamZero (Ye et al., 2026b), Cosmos Policy (Kim et al., 2026b), and ABot-M0.5 (Chen et al., 2026a). RoboCasa365 comparisons additionally include Azero-Robotics-1, WorldDreamer, GR00T N1.5/N1.6, GigaWorld-Policy (Ye et al., 2026a), RLDX-1 (Kim et al., 2026a), and PRTS (Zhang et al., 2026b), with external baseline scores from the leaderboard (RoboCasa Team, 2026), except Qwen-RobotManip, whose scores come from its technical report (Yuan et al., 2026a).

Video initialization comparisons use Wan2.2 (Team Wan et al., 2025), LongLive-2.0 (Chen et al., 2026b), and our LongLive2.0-Robot; GPT-6 Astra serves as the high-level planner in Appendix F.

F Atomic Execution and Compositional Planning

RoboCasa365 (Nasiriany et al., 2026) separates atomic household skills from their composition into multi-stage tasks, allowing us to examine Long-WAM as the execution foundation of a hierarchical robot system. We use a Human300-trained checkpoint with 2.4 seconds of visual context, then add GPT-6 Astra without further policy training. The planner grounds task goals into atomic subinstructions, selects execution-prefix lengths, and can issue bounded end-effector corrections; Long-WAM supplies the learned action chunks. Table 9 compares GPT-6 Astra alone, standalone and planner-augmented policies, and benchmark reference methods (RoboCasa Team, 2026).

Strong atomic execution. Long-WAM alone achieves 67.9% Atomic-Seen success, compared with 39.6% for $\pi_{0.5}$ and 31.5% for GPT-6 Astra alone, providing a strong physical skill foundation. Querying the same policy every 15 steps yields 84.4% without a planner, close to the hierarchical system's 85.6%.

Planning unlocks unseen skill compositions. With the policy checkpoint and visual context unchanged, adding the planner raises Composite-Seen success from 15.8% to 38.8% and Composite-Unseen success from 6.1% to 35.0%; Overall improves from 31.4% to 54.4%. The 15-step control reaches only 11.2% and 5.0% on the composite splits, so shorter execution intervals alone do not explain these gains.

Strong policies amplify agentic planning. GPT-6 Astra alone reaches 25.2% Overall success, and pairing it with $\pi_{0.5}$ reaches 30.5%, compared with 54.4% for Long-WAM + GPT-6 Astra. On unseen compositions, the Long-WAM hierarchy achieves 35.0%, exceeding both GPT-6 Astra alone (20.8%) and the $\pi_{0.5}$ hierarchy (19.0%). Planner augmentation yields a larger reported Overall gain for Long-WAM: 23.0 percentage points, versus 13.6 for $\pi_{0.5}$. These system-level comparisons highlight execution quality as a key complement to reasoning: Long-WAM supplies strong physical skills, while high-level planning extends their use to unseen compositions. This supports the division of labor in Appendix A, in which agentic planning and memory-informed execution jointly enable more capable long-horizon behavior.

G Discussion of Asynchronous Overlap Length

We evaluate Long-WAM (IDM) on RoboTwin 2.0 (Chen et al., 2025a), varying the trigger stride S with the execution horizon fixed at $R = 24$, giving overlaps $O = R - S$ of 12, 8, and 4 control steps. Table 10 reports Long-WAM's success rate and action continuity under these settings. The $S = 12$ result is also used for Long-WAM (IDM) in Table 6.

Continuity metrics. Overlap RMSE compares predictions aligned to the same absolute control steps within the overlap of O control steps. Let $\mathcal{C}$ contain the non-gripper action dimensions and $d_c = |\mathcal{C}|$. Using the action chunks defined in Section 3.1, we compute

$$E^k_{\text{ovlp}}(S) = \frac{\|\widehat{\mathbf{A}}_{t_k}[S:R,\mathcal{C}] - \widehat{\mathbf{A}}_{t_{k+1}}[0:O,\mathcal{C}]\|_F}{\sqrt{Od_c}}.$$

Slices are zero-indexed with exclusive upper endpoints. Jerk is a per-step proxy computed from measured non-gripper joint states q_t: we take the RMS of $q_{t+3} - 3q_{t+2} + 3q_{t+1} - q_t$ over windows that cross a chunk handoff, without dividing by Δt^3. Both metrics are computed per episode and then averaged equally over episodes with valid measurements, including successful and failed episodes.

Smoothness and decision frequency. Both RMSE and jerk decrease as the overlap shortens, with a larger reduction from 12 to 8 steps than from 8 to 4 steps. This trend motivates the short-overlap execution enabled by streaming VAE (Section 3.1). The success rate does not improve monotonically with smoothness: it rises from 94.2% to 95.0%, then falls to 94.3%. We interpret this pattern as a trade-off between action continuity and decision frequency. At a fixed control interval Δt, increasing S lengthens the nominal interval between inference decisions, $S\Delta t$, reducing the nominal decision frequency to $f_{\text{decision}} = 1/(S\Delta t)$ when no handoff waits occur. The robot still executes actions at the control rate $1/\Delta t$. Slower feedback can therefore offset the benefit of smoother chunk transitions. Minimizing discontinuity alone need not maximize task success.

Table 10: **Effect of asynchronous overlap length.** With $R = 24$, increasing the trigger stride S shortens the overlap O. RMSE and jerk decrease throughout the tested range, whereas the highest observed success rate (SR) occurs at $S = 16$. Stride and overlap are measured in control steps.

S	Overlap O	SR (%) ↑	RMSE ↓	Jerk ↓
12	12	94.2	0.0246	0.0436
16	8	**95.0**	0.0063	0.0156
20	4	94.3	**0.0058**	**0.0143**

Table 11: **Optimized IDM and CoD inference on RTX 5090.** V4/A4 and V2/A2 use four and two steps per expert, respectively; CoD shares the denoising schedule across experts. Latency includes observation VAE computation. Rates are the reciprocals of inference latencies, not robot control frequencies.

	V4/A4		V2/A2	
Method	Latency (ms) ↓	Rate (Hz) ↑	Latency (ms) ↓	Rate (Hz) ↑
Long-WAM (IDM)	107.4	9.3	81.8	12.2
Long-WAM (CoD)	90.9	11.0	65.7	15.2

H IDM VERSUS CO-DENOISING: CAPABILITY AND INFERENCE LATENCY

Long-WAM (IDM) retains video-first causal imagination: it predicts future visual latents independently of action tokens, then denoises actions conditioned on those predictions (Section 2.2). Co-denoising (CoD) instead updates future video and actions jointly, with bidirectional interaction at each denoising step. IDM provides the action expert with an explicit prediction of the future before generating actions, at the cost of sequential video and action inference. IDM achieves modestly higher reported average success rates on both LIBERO and RoboTwin 2.0 (Tables 1 and 2).

Latency protocol. We compare optimized end-to-end inference latency on NVIDIA GeForce RTX 5090, including the full observation VAE computation but excluding preprocessing, text encoding, and controller/IPC overhead. Both variants use video NVFP4, BF16 action compute and KV storage, and observed-video KV reuse. For CoD, the first joint step builds the history KV cache for reuse by later steps; future-video and action KV continue to be updated jointly. Each variant uses its own trained denoising schedule.

Reported latencies are the mean of the medians from two independent processes, each with five excluded warmups and 30 steady-state samples. The cumulative V4/A4 study in Table 7 uses the same aggregation and sample counts, excludes cold stabilization, and includes the full observation VAE and inference computation while excluding setup, text encoding, controller overhead, and IPC. Table 11 combines the IDM measurements from Table 5 with the CoD measurements. These timing runs are separate from the task-success evaluations above; the comparison uses each variant's trained schedule and cache implementation, so it does not isolate execution order as the sole source of the latency difference.

Capability and compute rate. CoD takes 90.9 ms with V4/A4 and 65.7 ms with V2/A2, compared with 107.4 and 81.8 ms for IDM (Table 11). IDM achieves higher success in the separate task evaluations, while its optimized inference reaches 9.3 and 12.2 Hz at the two budgets. The two-step configuration therefore fits within a 100 ms inference-compute budget. Online execution must also accommodate transfer and scheduling costs (Equation 4); asynchronous execution overlaps inference with ongoing robot motion. The real-world results in Section 4.4 provide separate evidence that IDM's video-first design supports responsive manipulation.

Observed-video KV reuse in IDM. Observed-video KV reuse changes token shapes. On RTX 5090, shape-specific dispatch extends the base quantizer's 588-token tile choice to the 392/196-token inputs produced by reuse. With the complete V4/A4 configuration, latency is 107.4 ms with reuse versus 112.6 ms without it (4.6%). Spark reaches 328.2 ms versus 394.9 ms without reuse (16.9%). With V2/A2, reuse gives no latency reduction on RTX 5090 (81.8 versus 81.3 ms), while

Table 12: **Context-dependent latency on RTX 5090.** End-to-end inference time per action chunk with the Long-WAM infrastructure. Context P counts preceding control intervals, corresponding to $P/20$ seconds of history; $P = 0$ retains the current observation.

Context P	History (s)	Latency (ms) ↓
0	0.0	74.6
48	2.4	107.4
96	4.8	138.3
192	9.6	204.5
384	19.2	341.0

latency on Spark decreases from 280.8 to 254.4 ms (9.4%). These comparisons hold the remaining configuration fixed within each device and denoising budget. They differ from the reuse-stage gains in Table 7, which are measured before subsequent device-specific tuning.

I Context-Dependent Inference Latency

Table 12 shows efficient long-context inference with our infrastructure on RTX 5090: increasing history eightfold, from 2.4 to 19.2 seconds, raises end-to-end chunk latency from 107.4 to 341.0 ms (approximately 3.2×). These results use the shared optimizations and device-specific tuning in Section 3.2. For online execution, streaming VAE additionally moves prefix encoding ahead of the inference trigger, reducing post-trigger preparation work (Section 3.1).

J Additional Real-World Deployment Visualizations

We provide detailed rollouts complementing the real-robot evaluation in Section 4.4. The montages retain the original overview and wrist-camera views, with red annotations marking failure events and green annotations marking successful outcomes. These selected examples illustrate execution behavior; aggregate success rates are reported in the main text.

J.1 Dynamic Composite Manipulation

Figures 9–11 show dynamic cup stacking on Unitree G1. Long-WAM coordinates grasping the blue cup, intercepting the moving green cup, and nesting it inside the blue cup. The baseline rollouts instead miss the green cup during interception.

J.2 Moving-Object Grasping at Different Speeds

Figures 12–14 compare conveyor grasping at four speeds. The baseline examples expose missed interceptions at higher speeds, whereas Long-WAM completes the illustrated grasp at every speed, consistent with the dynamic-control results in Section 4.4.

J.3 Long-Horizon Manipulation

Figure 15 shows Long-WAM executing three YAM tasks that require successive object interactions. The sequences illustrate task-directed progress through repeated pickup and placement, complementing the fast dynamic behaviors above. These tasks last over 40 seconds on average (Section 4.4).

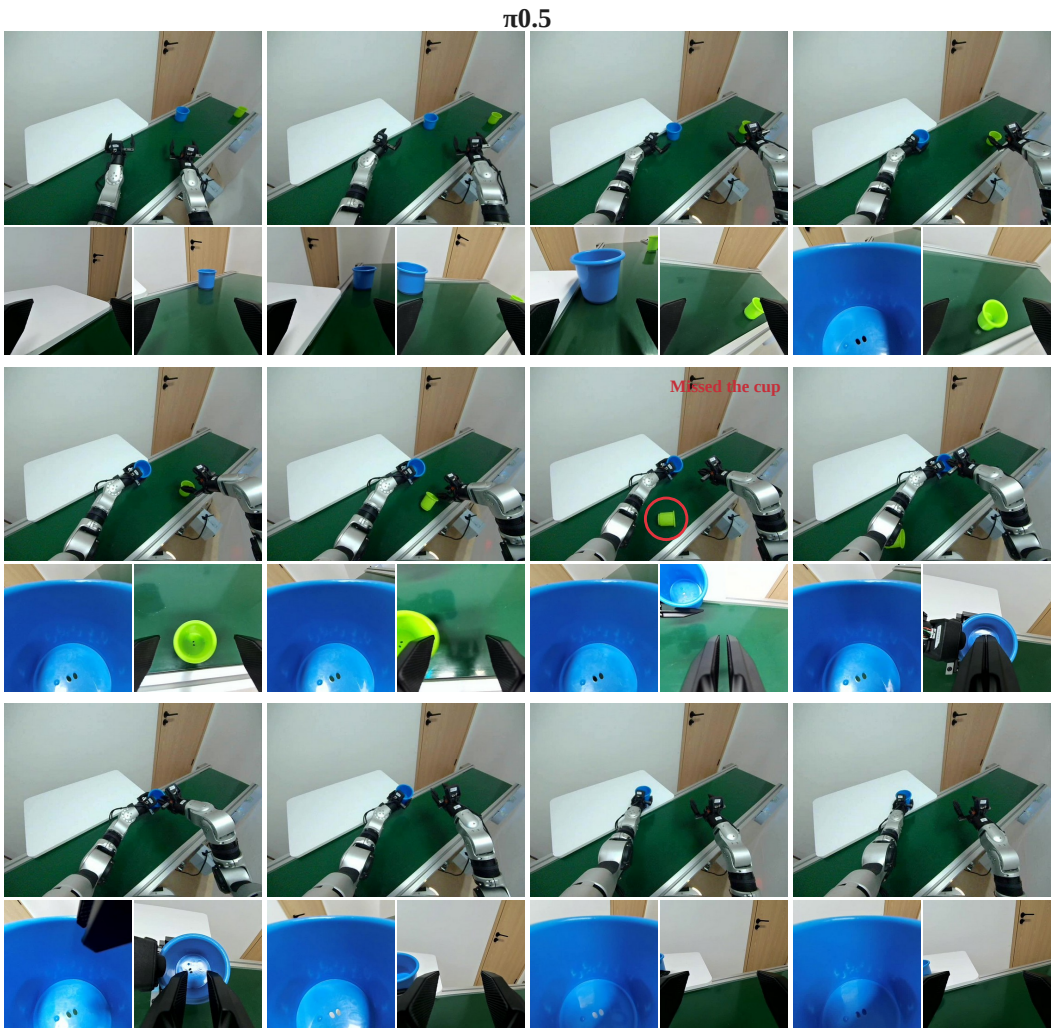

Figure 9: **Dynamic cup stacking with $\pi_{0.5}$.** Time progresses left to right within each four-column block, then continues in the next block below; each overview frame is accompanied by two wrist views. The policy grasps the blue cup but misses the moving green cup (red circle), leaving the composite task incomplete.

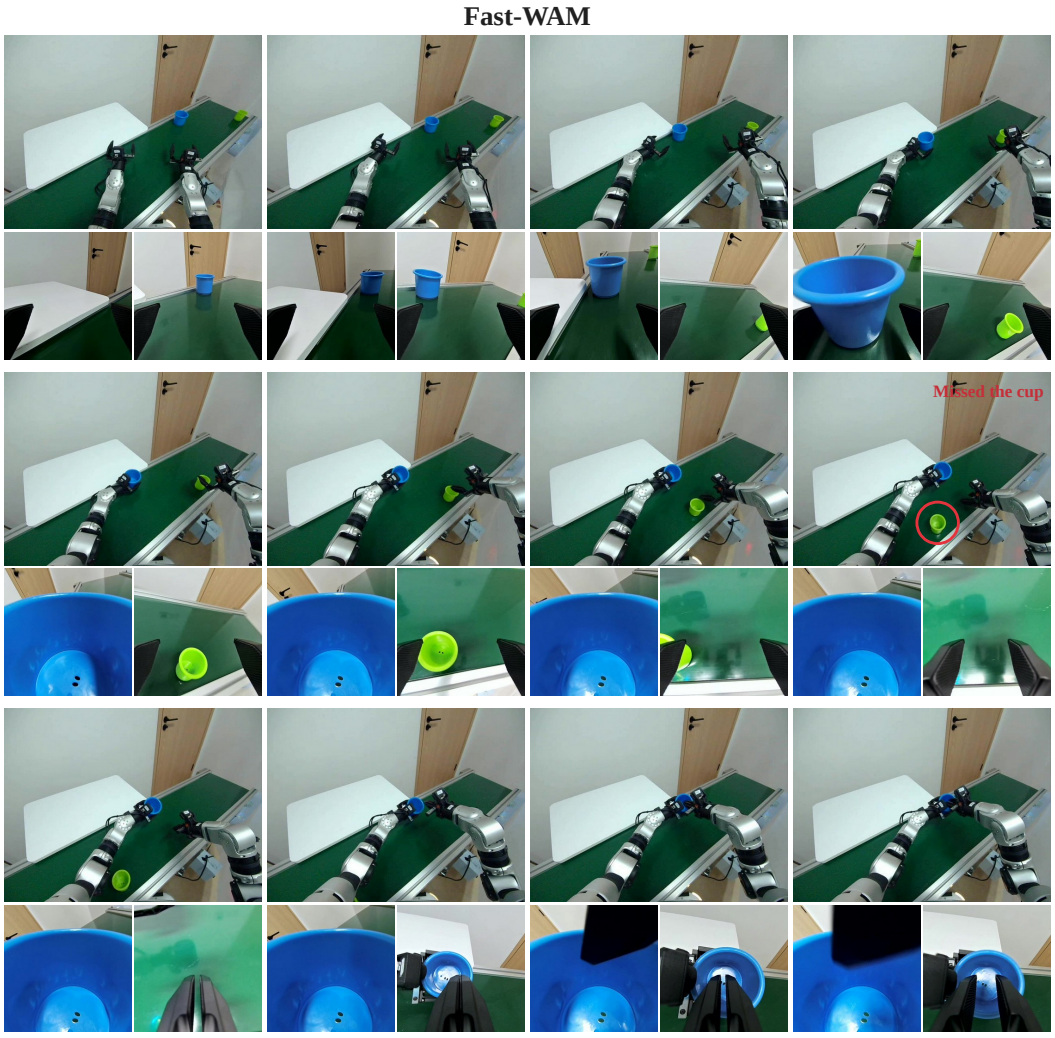

Figure 10: **Dynamic cup stacking with Fast-WAM.** The sequence follows the same reading order as Figure 9. After grasping the blue cup, the policy reaches toward the green cup but misses it as it moves along the conveyor. The subsequent frames show that the nesting stage is not completed.

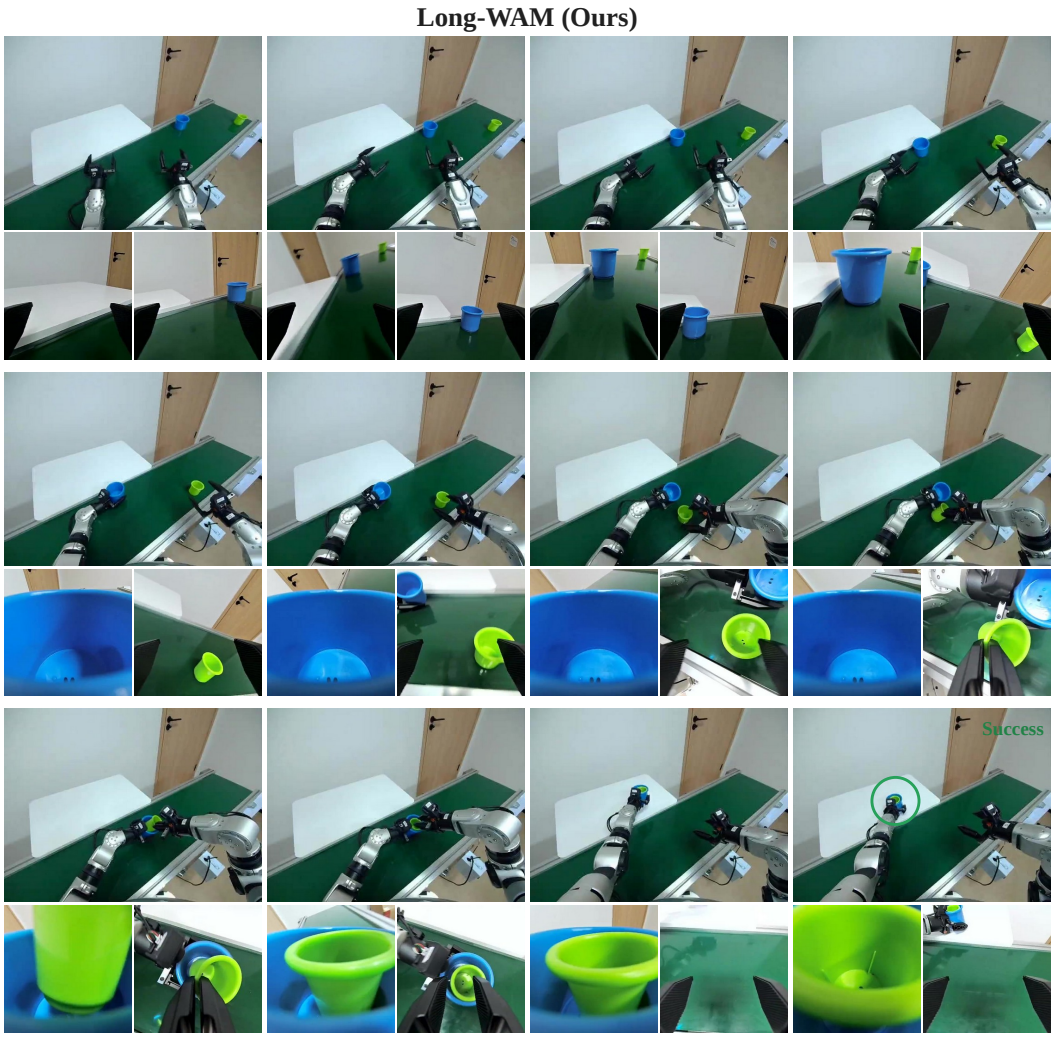

Figure 11: **Dynamic cup stacking with Long-WAM.** Long-WAM grasps the blue cup, intercepts the moving green cup, and places the green cup inside the blue cup. The overview and wrist views reveal the transition from interception to alignment and insertion, illustrating coordinated execution across the stages of this dynamic task.

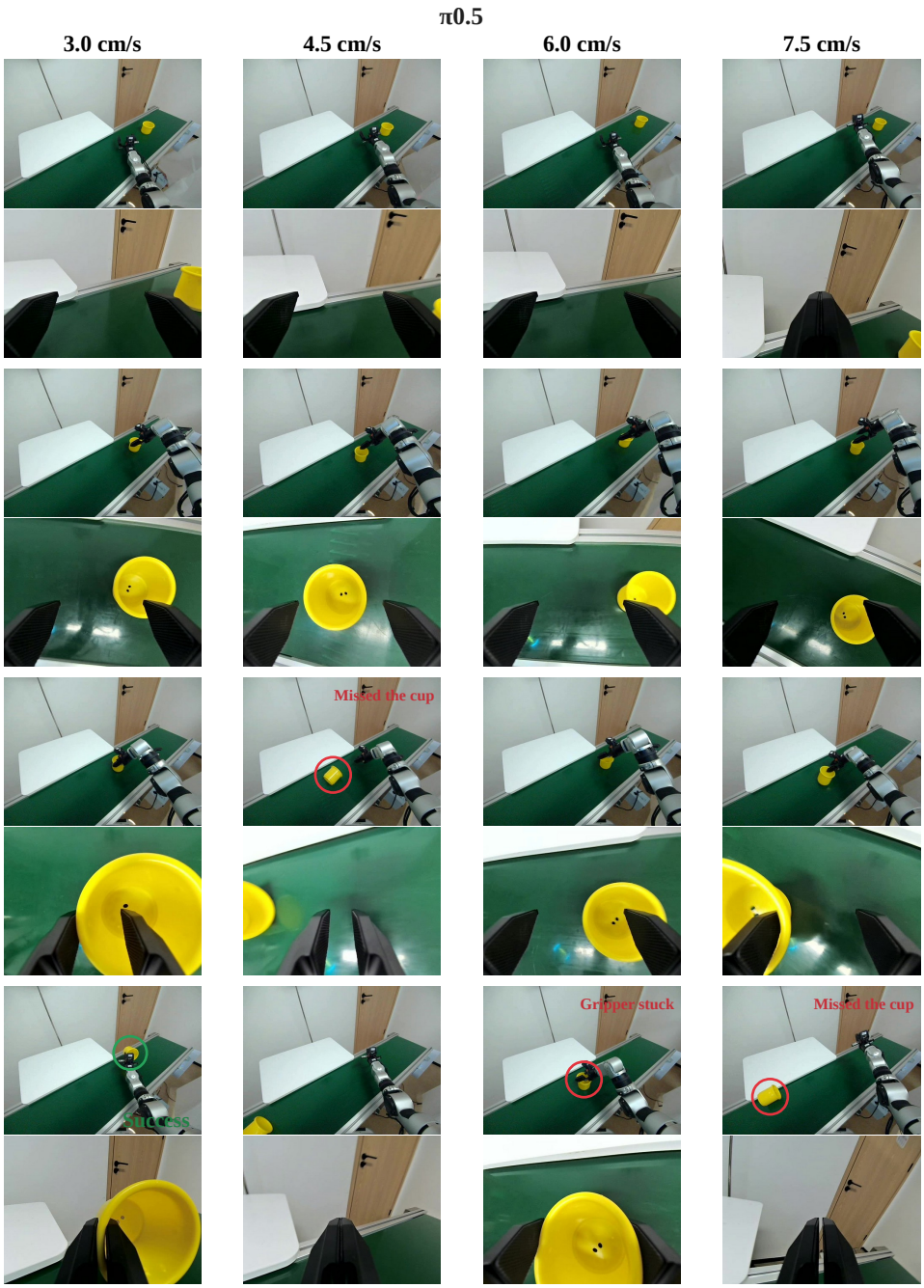

Figure 12: **Moving-object grasping with $\pi_{0.5}$.** Columns show 3.0, 4.5, 6.0, and 7.5 cm/s; time advances downward through paired overview and wrist views. The selected rollout succeeds at 3.0 cm/s, misses the cup at 4.5 and 7.5 cm/s, and exhibits the annotated gripper-stuck failure at 6.0 cm/s.

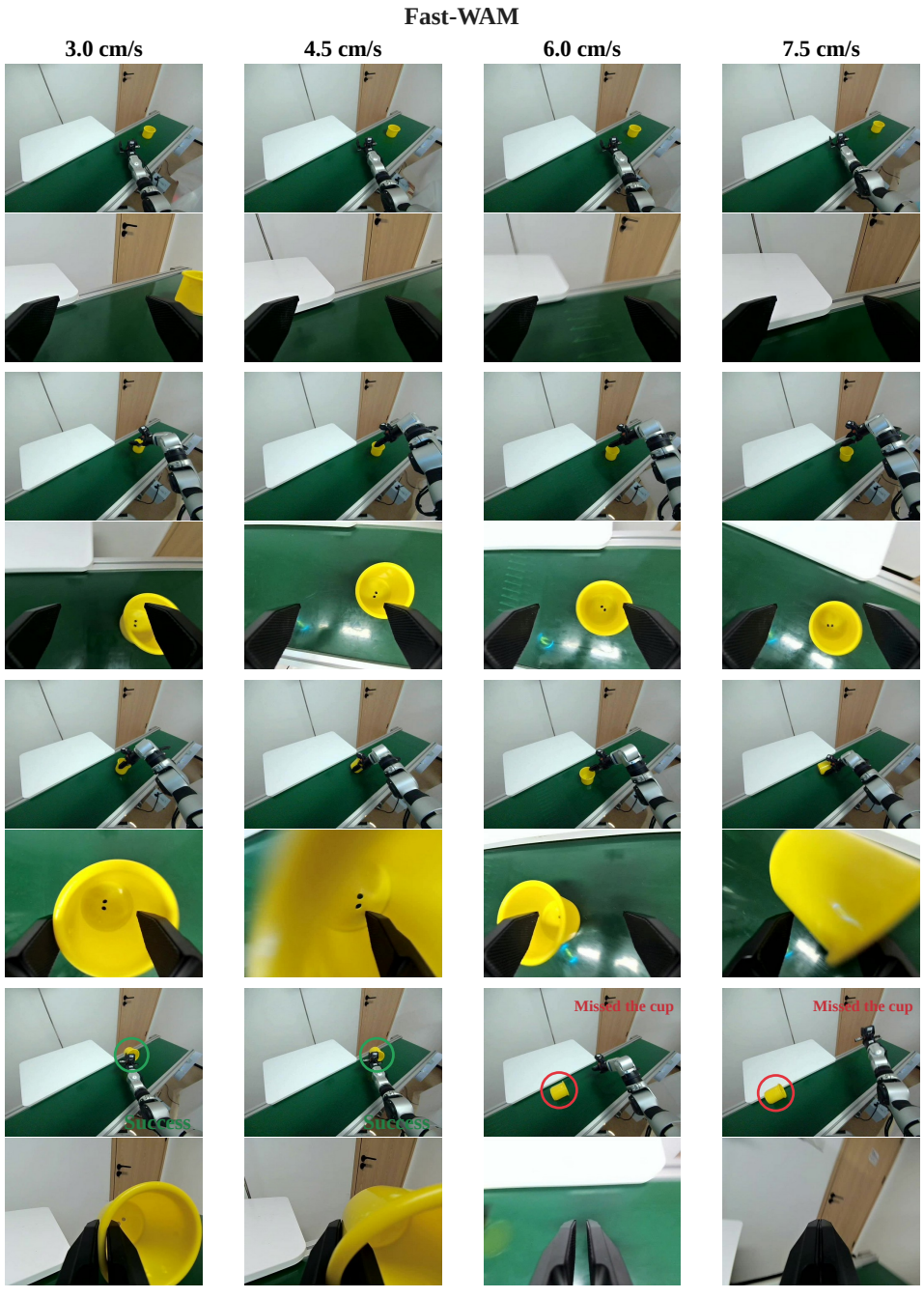

Figure 13: **Moving-object grasping with Fast-WAM.** Columns and temporal ordering match Figure 12. The selected rollouts succeed at 3.0 and 4.5 cm/s but miss the cup at 6.0 and 7.5 cm/s. The wrist views show the target moving beyond the gripper before a secure grasp is established.

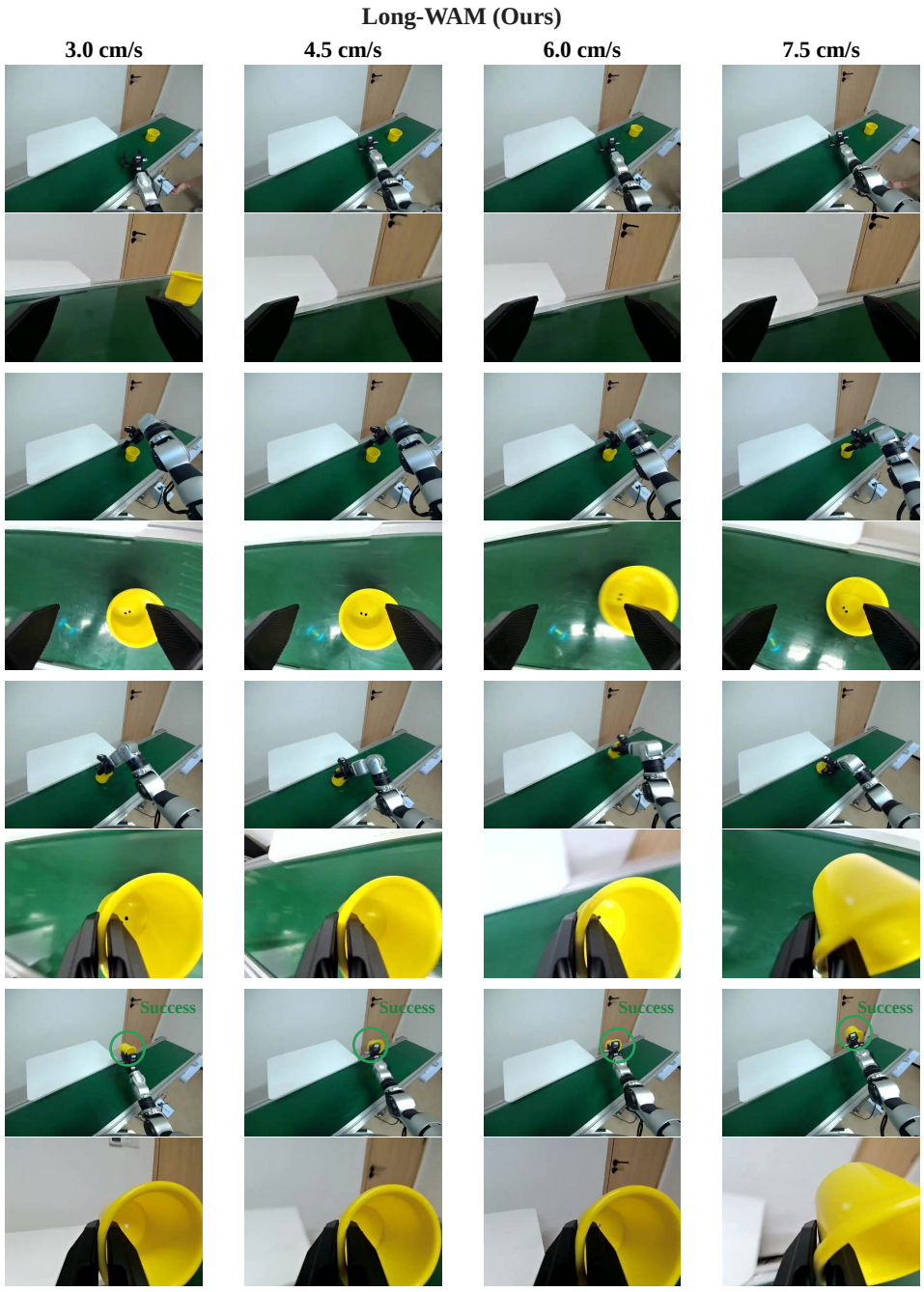

Figure 14: **Moving-object grasping with Long-WAM.** The illustrated rollouts complete the grasp at all four conveyor speeds, including 6.0 and 7.5 cm/s. Successive wrist views show the moving cup entering the gripper and being retained after closure, illustrating responsive interception under progressively tighter timing constraints.

Long-WAM (Ours)

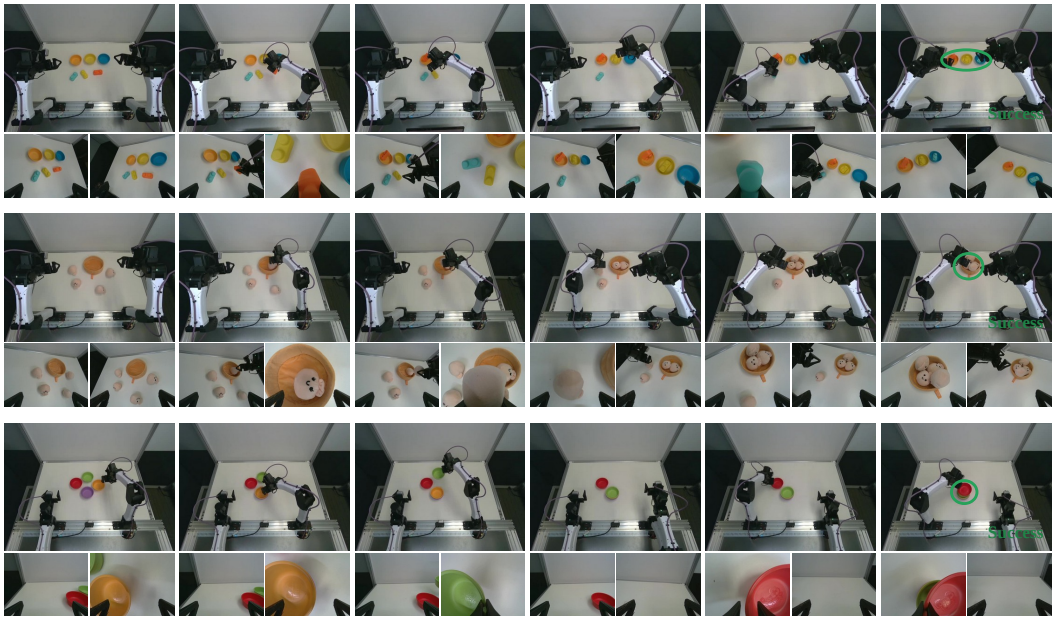

Figure 15: **Long-horizon manipulation with Long-WAM on YAM.** Top to bottom: brick sorting by color, placing dumplings in a pan, and stacking bowls. Each task contains six chronological snapshots, read left to right, with paired wrist views below each overview frame. The final snapshots show successful task configurations across all three tasks.