WALL-WM: Carving World Action Modeling at the Event Joints

X Square Robot Team

WALL-WM is a World Action Model that shifts video-action learning from chunk-centric optimization to event-grounded Vision-Language-Action pretraining, using semantically coherent action events as the atomic unit of learning. Existing WAMs commonly initialize from multimodal or video foundation models and then optimize fixed-length action chunks conditioned directly on the current observation and instruction. Although convenient, this chunk-centric formulation creates a fundamental granularity mismatch. Language describes semantic goals and events, vision evolves through continuous scene dynamics, and actions operate at control-level timescales; forcing all three into the same fixed-length prediction window turns VLA training into short-horizon correlation fitting. This not only underuses the pretrained visual-semantic prior, but can actively overwrite it with chunk-specific action shortcuts, weakening compositionality and long-horizon generalization. WALL-WM addresses this mismatch by organizing both supervision and data around semantic events. Specifically, it pairs event-grounded VLA pretraining with a data ecosystem built from event-level captions and cluster-balanced sampling, enabling scalable learning over diverse behaviors, scenes, and task structures. From the same event-pretrained backbone, WALL-WM supports two complementary inference modes. The event mode consumes next-event descriptions and enables variable-length execution chunks, while the unified mode uses a VLM with Staircase Decoding to condition conventional fixed-length chunk inference while preserving a gradient-continuous VLA path. Together with Muon-optimizer-based large-scale pretraining infrastructure, WALL-WM provides a practical scale-up recipe for general-purpose WAMs. Experiments show that WALL-WM generalizes broadly across language, scenes, and tasks, achieving state-of-the-art performance in large-scale real-world generalization evaluation.

"Carve nature at its joints."

— after Plato, *Phaedrus* 265e

Date: May 29, 2026
Code: https://github.com/X-Square-Robot/wall-x

1 Introduction

Recent progress in embodied foundation models (97, 41, 9, 36, 8, 14, 17) has increasingly been driven by large-scale priors inherited from multimodal understanding models and video foundation models (20, 37, 13, 95). In this report, we use VLA to broadly denote embodied foundation models that predict actions from visual-language inputs, and WAMs to denote world-action models that explicitly couple future observation modeling with action prediction.

Most existing embodied foundation models adapt these priors by predicting fixed-length action chunks from the current observation and language instruction. This chunk-centric formulation is effective and convenient, but it hides a structural mismatch: it cuts embodied dynamics by an external clock, while language, vision, and action evolve at different semantic and temporal scales. Language specifies goals and events; vision evolves through continuous scene dynamics; action operates at control-level time scales and is sensitive to contact, timing, and small perturbations. Thus, adapting pretrained visual-semantic priors to embodied control is not merely a fine-tuning problem. It is first a question of where to place the atomic unit of video-action learning.

This distinction is the starting point of WALL-WM:

Fixed chunks cut by clock; semantic events cut by embodied dynamics.

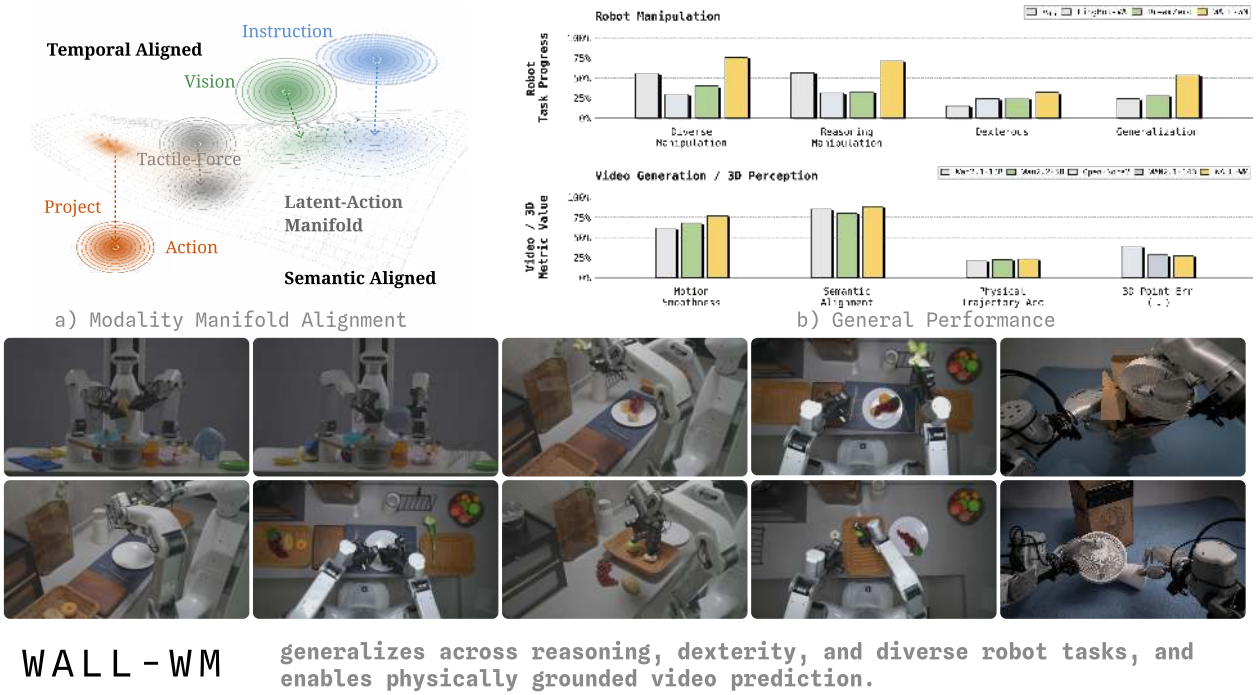


Figure 1 Conceptual illustration of modality hierarchy and WALL-WM's general performance. *Left:* a stylized alignment landscape over semantic abstraction and spatial-temporal precision. Text provides coarse semantic alignment, vision provides denser spatial-temporal grounding, and action requires fine-grained contact-sensitive precision. Tactile-force input, when available, is treated as an optional contact-rich signal rather than a required modality. *Right:* WALL-WM shows clear advantages across manipulation-task performance and video-generation metrics, indicating that event-grounded pretraining improves both executable control and future-observation modeling. *Bottom:* representative real-robot task snapshots provide illustrative examples of the physical tasks.

The rest of the argument follows from this mismatch. The central challenge is not multimodal fusion in the usual sense, but *geometry-preserving alignment*. Text, vision, and action do not share the same notion of neighborhood: semantically similar instructions may induce different visual trajectories, while visually similar states may require incompatible control responses. As illustrated in Fig.1, text provides coarse semantic alignment, vision supplies denser spatial-temporal grounding, and action requires the finest local precision. A useful WAM must connect these modalities without flattening them into a single undifferentiated embedding space.

Video is the natural scaffold between language and action. Internet-scale video pretraining captures rich visual dynamics that would otherwise have to be learned from embodied interaction alone(37, 20). Video is also semantically structured enough to align with language at event boundaries, yet temporally dense enough to expose the timing, transitions, and state changes needed for action execution. Future video shares causal temporal structure with action, making visual-to-action grounding and inverse dynamics possible (20, 43). In this sense, video offers an embodiment-light bridge from high-level semantic intent to low-level execution.

Turning this scaffold into a WAM is not a short adaptation stage. It is a *prior-preserving lift*: the model must inherit the semantic and temporal structure learned from large-scale video, while acquiring the controllability, contact sensitivity, and causal grounding required for embodied action. Simply appending an action decoder can attach actions to a visual prior, but it does not determine the unit at which the prior should become executable. Without such a unit, joint optimization can collapse toward the most data-rich modality or overwrite useful visual-semantic structure with short-horizon action correlations.

This lift imposes two requirements. First, a WAM must *preserve the video prior*: video generation models favor semantic invariance, visual plausibility, and temporal smoothness, whereas embodied control requires sensitivity to action-induced divergence and contact transitions (86, 44). Second, a WAM must provide *temporal grounding*: language instructions often describe global tasks or semantic events, while observations and actions

unfold over many frames and control steps. Fixed-length chunks are poorly matched to both requirements. They can be too short to contain a complete semantic event, yet too long to preserve clean causal separation between context and prediction target.

The same reasoning clarifies what must be lifted from a video prior into a general-purpose WAM:

T1. Reasoning: convert global instructions and task progress into event-structured intent.

T2. Visual prediction: preserve the caption-to-video inductive bias while making future observations controllable by executable events.

T3. Fine manipulation: expose timing, contact transitions, and local state changes required for action execution.

These layers cannot be obtained by stacking a VLM, a video model, and an action head as independent modules. They require a training regime whose alignment unit is simultaneously meaningful to language, visible in video, and executable through action.

This gives three design principles for the alignment unit:

- **Geometry preservation:** connect language, video, and action without collapsing their native structures into one shared space.
- **Prior preservation:** remain compatible with the caption-to-video structure inherited from video foundation models.
- **Executable causality:** provide a prediction target with clear temporal support, while allowing duration to follow the task rather than a fixed clock.

These principles rule out the fixed-length action chunk as the fundamental unit. A chunk is convenient for batching and deployment, but it is not a natural object shared by language, video, and action. It may cut through the middle of a semantic behavior, merge multiple behaviors into one target, or require historical context merely to determine what the chunk is supposed to mean.

WALL-WM therefore replaces the fixed-length chunk with an *action-grounded semantic event*: a temporally coherent segment of executable behavior, such as reaching, grasping, lifting, moving, or placing, that is expressible in language, observable in video, and realizable through action. Unlike a fixed temporal chunk, which follows an external clock, an action-grounded semantic event begins and ends when the underlying executable behavior changes. It therefore satisfies the principles above: language names the event, video grounds its spatial-temporal evolution, and action realizes it through control.

WALL-WM instantiates this principle through event-grounded WAM pretraining. Event captions are paired with corresponding video and action segments, and the model is trained to denoise future video and action over event-aligned intervals. This does not merely use events as auxiliary conditions; it grounds the training problem itself at the event level. The result is a prior-preserving route from video foundation models to executable world-action models.

At inference time, the same event-pretrained backbone supports two complementary modes:

Event mode. The system rolls out in event space. A VLM, human, or agent proposes the next-event description, and WALL-WM executes the corresponding variable-length video-action segment before observing the next state. This mode follows the natural duration of the task rather than a fixed control horizon.

Unified mode. Conventional fixed-length chunk inference is retained, but the chunk is no longer conditioned on a raw global instruction alone. A VLM with Staircase Decoding supplies event-structured latent reasoning over task progress, producing a latent event representation that guides the next local chunk while preserving a gradient-continuous VLA path.

Fig.2 summarizes these two schemes, and Sec.3 provides the detailed formulation.

This report presents WALL-WM as both a model instance and a training roadmap for prior-preserving scale-up of WAMs. Its main components are:

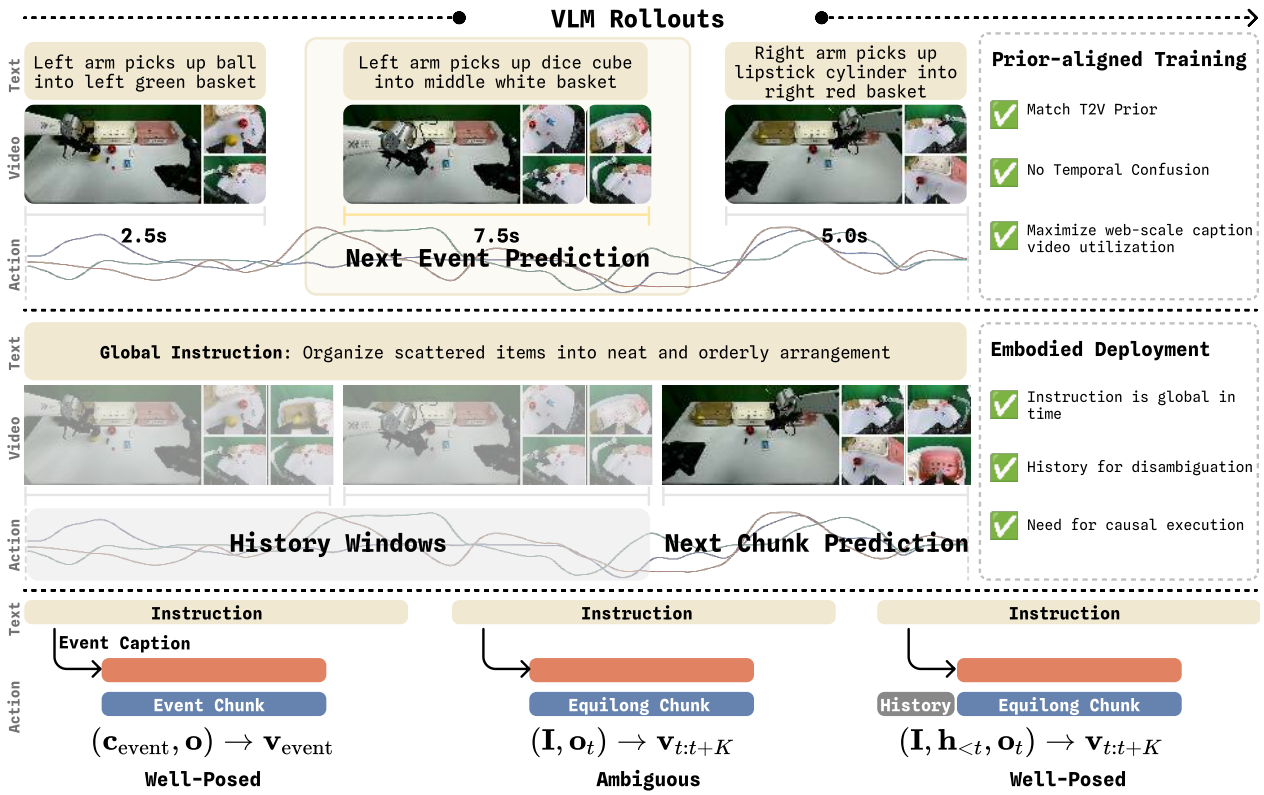


Figure 2 Next-event training and equilong-chunk schemes. In prior-aligned training, the event caption, event video, and event action describe the same semantic interval, giving a well-posed caption-to-video/action target. In equilong-chunk mode, a global instruction alone is ambiguous for a local chunk; adding history windows restores a well-posed next-chunk prediction problem.

- **Event-grounded WAM pretraining.** WALL-WM treats action-grounded semantic events as the atomic training unit, pairs them with event captions, and trains a video-action denoiser that preserves the inherited video prior while turning event-aligned visual evidence into executable action.
- **Two inference modes from a single event-pretrained backbone.** Event mode performs event-space rollout with variable-length executable segments. Unified mode supports fixed-length chunk prediction by using VLM-based Staircase Decoding to generate event-structured latent reasoning while maintaining a gradient-continuous VLA path.
- **Scale-up infrastructure for event-grounded embodied modeling.** WALL-WM combines an event-grounded data ecosystem, cluster-balanced sampling, and Muon-optimizer-based large-scale pretraining infrastructure to support scalable WAM training across diverse behaviors, scenes, and tasks.

WALL-WM demonstrates broad generalization across language instructions, scenes, and tasks, achieving strong performance in large-scale real-world generalization evaluation. Overall, WALL-WM connects text, video, and action through action-grounded semantic events. We position WALL-WM less as a short adaptation of a video foundation model and more as a prior-preserving scale-up methodology for the next generation of embodied foundation models.

2 Related Work

2.1 Vision-Language-Action Models

A growing line of embodied foundation models, commonly referred to as Vision-Language-Action (VLA) policies, extends pretrained vision-language models with action interfaces that map visual observations and natural-

language instructions to executable motor commands (97, 41, 70, 9, 36, 8, 14, 77, 68). By inheriting web-scale semantic priors from VLM pretraining, these models exhibit strong generalization across objects, scenes, and language instructions, and offer a unified alternative to modular perception–planning–control pipelines. Recent work has explored a broad range of design choices within this observation-to-action paradigm, including efficient action tokenization and chunking for long-horizon control (90, 60, 41), diffusion- or flow-matching-based action experts (16, 45, 74, 9), lightweight and data-efficient backbones (75), latent-action pretraining and cross-embodiment routing (82, 11), knowledge-insulated fine-tuning that preserves pretrained priors during adaptation (19), spatially enhanced or 3D-aware policy representations (61, 56), and visual chain-of-thought reasoning interleaved with action generation (87, 89). Despite their broad semantic capabilities, existing VLAs still face systematic limitations. First, their underlying VLMs are pretrained predominantly on static image–text data (97, 9), so even large-scale supervised fine-tuning on teleoperation datasets (41, 8) primarily learns action imitation rather than an explicit model of how the physical world evolves under intervention. Second, most VLAs formulate control as a reactive observation-to-action mapping without action-conditioned future prediction or an explicit temporal-dynamics prior (45, 14). Third, generalization to genuinely novel motions, skills, embodiments, or environments often still requires substantial task- or robot-specific adaptation data (36, 17). These limitations have motivated a parallel line of research that grounds policies in generative models of how the world evolves, which we review next.

2.2 Generative Embodied World Models

Generative world models have recently emerged as a promising paradigm for embodied AI, where agents learn to predict future states of the physical world and use such predictions for planning and control. Early model-based reinforcement learning methods such as PlaNet, Dreamer, and DreamerV3 (28, 27, 29) learn compact predictive states for control, while the JEPA family, including V-JEPA and LeWorldModel (2, 54), shows that forecasting in feature space can capture meaningful temporal and physical structure without explicit pixel reconstruction. More recent embodied world models further leverage generative video and world-action modeling objectives to learn robot–environment dynamics from large-scale heterogeneous data. These models use predicted future frames or intermediate visual representations as plans, from which executable actions are recovered through inverse-dynamics models (20), intermediate-feature action decoders (59), dense correspondence (43), planning-oriented trajectory decoding (21, 94), or synthesized demonstrations transferred from human videos and novel scenes (6, 5, 15, 37, 51). Recent work further suggests that video generators encode useful 3D and interaction priors (32, 42, 93, 78, 46), motivating their use as embodied dynamics models. Along this direction, LaDi-WM, AdaWorld, Motus, LDA-1B, and MotuBrain (33, 23, 7, 53, 69) explore latent diffusion, structured visual forecasting, latent-action conditioning, and unified video–action modeling, while LingBot-VA (44) and related unified denoising architectures (26, 22, 95, 13, 83, 76, 12, 88, 47) jointly model future prediction and action generation. More recent methods such as Fast-WAM (86) further improve inference efficiency by avoiding explicit video decoding or future imagination at test time. Overall, these advances indicate that explicit future-state modeling can improve sample efficiency, robustness, and generalization.

2.3 Latent Reasoning

A complementary line of work routes chain-of-thought (CoT) reasoning through compact *latent* representations rather than emitting full textual reasoning traces, motivated by inference efficiency and the ability to explore semantic-level reasoning trajectories beyond discrete token sequences (30, 24, 38, 39, 50, 92). LaDiR (38) encodes reasoning steps into blocks of continuous "thought tokens" via a VAE and refines them with a latent diffusion model; its reinforcement-learning extension (39) further optimizes latent trajectories to preserve solution diversity. Recent VLA methods further extend this idea to physically grounded latent reasoning: LaST_0 (49) introduces a latent spatio-temporal CoT that captures future visual dynamics, 3D structure, and proprioceptive states for robotic manipulation; LaST-VLA (52) distills geometric constraints and world-model foresight into latent spatio-temporal representations for autonomous driving; and LaRA-VLA (3) progressively transfers textual and visual CoT supervision into latent reasoning dynamics for efficient action generation. In contrast, WALL-WM uses a staircase parallel decoder to inject and propagate latent CoT tokens through staggered depths of the VLM backbone. This design preserves as much of the pretrained VLM's hierarchical visual-linguistic priors and causal computation structure as possible, while amortizing the layer cost of latent reasoning.

3 Architecture Design: Event-Centric World Action Modeling

3.1 Overview

As summarized in Fig. 3, WALL-WM instantiates *event-centric* world action modeling through a prior-aligned, multimodal pretraining stack: a video tower inherited from a Wan Series text-to-video model (72) and a randomly initialized action DiT are layer-coupled, with pretrained encoders held fixed and cross-modal alignment learned through layer-wise video-action couplings. Pretraining is organized at the *event* level: each sample is an atomic $(\mathbf{V}_e, \mathbf{a}_e)$ event carved out of a long-horizon episode rather than a fixed-length chunk, and the model is trained to denoise event-aligned futures from the current observation. At each block, the action stream cross-attends to the matched video features without modifying the video stream.

Formally, WALL-WM models $p_\theta(\mathbf{V}_e, \mathbf{a}_e \mid \mathbf{V}_0, \mathbf{s}, c_e)$, where $\mathbf{V}_0$ is the current multi-view observation (one keyframe per camera), $\mathbf{s}$ the current proprioceptive state, $(\mathbf{V}_e, \mathbf{a}_e)$ the event-aligned future multi-view video and end-effector trajectory (both with event-dependent length), and c_e the per-event caption that describes the same action-grounded semantic event. The remainder of Section 3 follows the two paradigms of Fig. 2: *next-event prediction* with the *event-centric* layout in subfigure (A) for pretraining and event-mode inference, and *next-chunk prediction* with the *observation-centered* layout in subfigure (B) for unified-mode deployment. First, Sections 3.2 and 3.3 build the shared video-action denoiser from event-centric pretraining through layer-wise coupling and temporal alignment, extending to the history-augmented observation-centered window. Second, Section 3.4 attaches inference-time language pathways to the same event-pretrained backbone: a vision-aware VLM bridge supports next-event conditioning in event mode and instruction grounding in unified mode, while Staircase decoding supplies event structure for fixed-length chunk prediction without token-by-token autoregression.

3.2 Multi-View Visual World Events Modeling

The video tower inherits the Wan single-view DiT and extends it to *multi-view, multi-embodiment* video generation. The inherited within-view computation stays in place; we graft three additions onto the same backbone: (i) *multi-view adaptation from single-view priors*, which runs rearranged cross-view self-attention over per-frame multi-view tokens and merges the result back through a zero-initialized output projector; (ii) *Camera RoPE*, which gives each camera a learnable, calibration-free identity in that cross-view branch so the same DiT can operate across heterogeneous multi-embodiment camera setups; and (iii) *cross-view geometric masking*, a complementary training-time pair that strengthens cross-view geometric consistency. The tower is then trained on event latents under Wan-style v-prediction flow matching (31, 64, 48).

Multi-View Adaptation from Single-View Priors. When $N_v > 1$, each DiT block runs an additional cross-view branch after the usual within-view Wan self-attention. Each camera stream still enters as its own batch item; inside a view-attention block we regroup hidden states frame by frame, concatenate all spatial tokens from the N_v cameras at each latent frame into one sequence, and run self-attention on that joint layout—with rotary codes that also encode the camera axis (Camera RoPE, next paragraph). The attention weights are initialized from the block's within-view self-attention; its output is passed through a zero-initialized projector and added back to the per-view stream under an AdaLN gate. Let $\mathbf{h}_i^V$ denote the hidden states at DiT block i (in the per-view Wan layout), $\mathrm{CrossViewAttn}_i(\cdot)$ the rearranged cross-view self-attention above, W_{view} its zero-initialized output projector, and g_i the AdaLN gate on that branch; then

$$\mathbf{h}_i^V \leftarrow \mathbf{h}_i^V + g_i\, W_{\text{view}}\, \mathrm{CrossViewAttn}_i(\mathbf{h}_i^V), \qquad W_{\text{view}} \text{ initialized to } 0. \tag{1}$$

Because W_{view} starts at zero, this branch contributes nothing at initialization and cross-view exchange turns on only as the projector learns during training. The within-view Wan stack is otherwise unchanged, so pretrained appearance and language-alignment behavior are preserved while cross-view exchange is learned on top.

Camera RoPE. To support large-scale multi-embodiment training, Camera RoPE gives each view a learnable rotary identity without feeding calibration to the model at runtime. We extend RoPE (67) with a view axis, partitioning each head's frequency bank over (f, h, w, view); the view rotation is produced from a learnable per-view embedding shared across all view-attention layers. Adding or removing a camera therefore only changes the embedding table. The rotary code tells the network which camera each token came from; during training, the sight-cone mask below further specifies which other tokens it may plausibly correlate with.

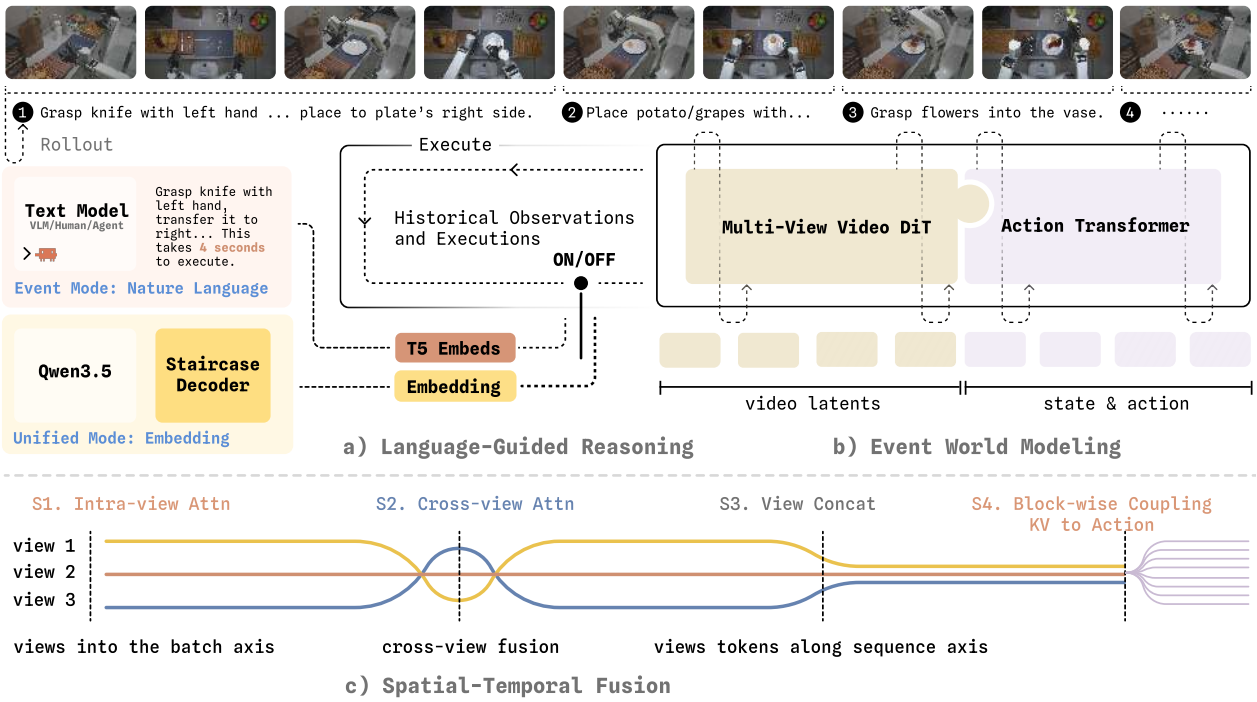


Figure 3 Overall framework of WALL-WM. WALL-WM implements *event-centric* world action modeling as a layer-coupled video-action denoiser: given the current multi-view observation and a next-event instruction, it jointly denoises future video latents and the corresponding end-effector trajectory. The figure is organized as three subfigures. (a) supplies instruction: event mode routes next-event language into the shared text conditioner (T5 (62) embeds in the figure), whereas unified mode routes through a Staircase decoder to CoT latents. (b) is the event-centric world model—a Multi-View Video DiT denoises video latents and an Action Transformer denoises actions; *Execute* and *Rollout* close the control loop above. (c) summarizes the spatial-temporal fusion that threads N_v=3 camera streams through the stack: view folding that preserves the inherited within-view self-attention with its 3D RoPE positional encoding (S1), cross-view mixing in each DiT block (S2), token-axis `ViewConcat` (S3), and one-way block-wise coupling of fused video keys/values into the action tower at every depth (S4).

Cross-View Geometric Masking. View attention alone can mix tokens across cameras even when their patches have no shared field of view, and when co-visible regions *do* exist the network often recovers them through the shorter temporal path inside a single view, so cross-view attention is either abused as a generic feature mixer or left under-trained on the pairs that matter. At training time, we address both failure modes with a complementary pair of geometry-aware masks built from the same per-robot calibration; both are dropped at inference so rollout stays calibration-free.

Sight-cone attention masking. For two video tokens $u=(v_u, h_u, w_u)$ and $u'=(v_{u'}, h_{u'}, w_{u'})$ on the same latent frame, we say the pair is *co-visible* if the viewing frustums of their related patches in the original videos intersect (there could be observations of the same region from different viewing angles in pixel space). For computational convenience, we model each frustum as a cone $C(u) = (\mathbf{p}_0(u), \hat{\mathbf{v}}(u), \gamma(u))$ with apex at the camera center, axis $\hat{\mathbf{v}}$ toward the patch center, and half-apex angle γ initialized to tightly enclose the patch (optionally scaled by $l \geq 1$). Cone parameters follow per-robot extrinsics, intrinsics, and distortion; intersections are tested in parallel within a depth-of-field band $[d_{\min}, d_{\max}]$:

$$\mathbf{p}(u,t) = \mathbf{p}_0(u) + t\hat{\mathbf{v}}(u), \tag{2}$$

$$(t_1, t_2) = \arg\min_{(t_1,t_2)} \|\mathbf{p}(u,t_1) - \mathbf{p}(u',t_2)\|_2, \tag{3}$$

$$\hat{t}_1 = \text{clamp}\,(t_1, d_{\min}, d_{\max}), \quad \hat{t}_2 = \text{clamp}\left(\arg\min_{t_2} \left\|\mathbf{p}(u,\hat{t}_1) - \mathbf{p}(u',t_2)\right\|_2, d_{\min}, d_{\max}\right), \tag{4}$$

$$C(u) \text{ intersects } C(u') \iff \left\|\mathbf{p}(u,\hat{t}_1) - \mathbf{p}(u',\hat{t}_2)\right\|_2 \leq \hat{t}_1\gamma(u) + \hat{t}_2\gamma(u') \quad \text{(approximately)}. \tag{5}$$

This yields a binary mask

$$\mathcal{M}_{\text{sc}}[u, u'] \;=\; 1 \;\iff\; C(u) \text{ intersects } C(u'). \tag{6}$$

We add $(1 - \mathcal{M}_{\text{sc}}) \cdot (-\infty)$ as an attention bias in every view-attention block, forbidding cross-view routing across geometrically incompatible patches; the mask is computed once per sample and reused across depth. At inference the bias is dropped, recovering unmasked attention with Camera RoPE. $\mathcal{M}_{\text{sc}}$ closes the loop with Camera RoPE above: rotary codes identify *which* camera each token came from, and the sight-cone mask identifies *which other tokens* it may correlate with.

Tube patch masking. The second mechanism creates an explicit *demand* for cross-view attention. With probability p_{tube} we pick a view v^* and a $k{\times}k$ spatial window with $k{\in}\{l_{\min}, \ldots, l_{\max}\}$, uniformly sampled inside $U{=}\{u \mid \exists u', v_{u'}{\neq}v^*, \mathcal{M}_{\text{sc}}[u, u']{=}1\}$ so recovery from other views is possible. The resulting *tube*—the same spatial window across all latent frames of v^*—is masked in the noised input $\mathbf{z}_t^V$ by replacing the tokens with pure noise; with nested probability $p_{\text{tube}}^{\text{cond}}$ the same tube is also masked on the conditioning channel $\mathbf{y}$. The reconstruction target is unchanged, but the masked tube has no within-view temporal shortcut, so recovery must route through the other $N_v{-}1$ views. Non-trivial $(p_{\text{tube}}, p_{\text{tube}}^{\text{cond}})$ are used when hardening cross-view correspondence; how the masked tube enters the video loss is specified in the flow-matching paragraph below.

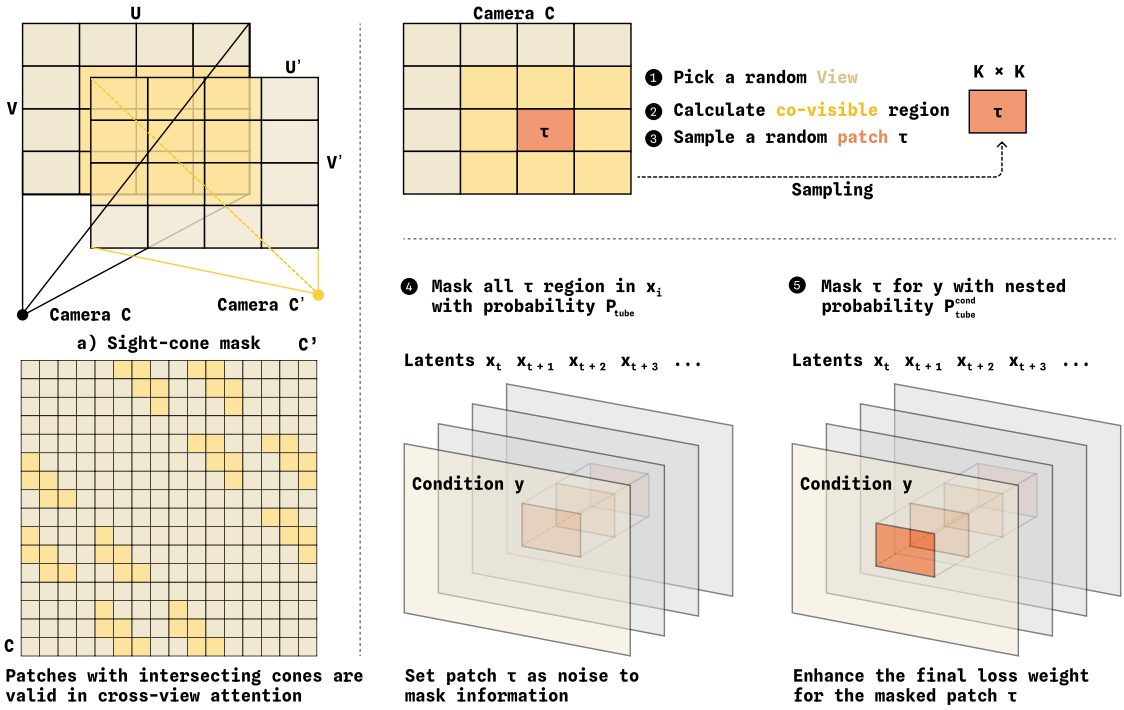


Figure 4 Cross-view masking in WALL-WM's view attention. *(a) Sight-cone mask.* A token pair (u, u') is allowed to attend if and only if their back-projected sight cones share a 3D region in front of both cameras; the resulting binary mask $\mathcal{M}_{\text{sc}}$ is added as a $(1 - \mathcal{M}_{\text{sc}})(-\infty)$ attention bias, with intra-view pairs always allowed (matrix diagonal). *(b) Tube mask.* A spatial window on one view v^* is masked across all latent frames—and, with nested probability, also on the InP conditioning channel $\mathbf{y}$—leaving cross-view attention to the other views as the only path that can recover the masked content; the in-tube v-prediction loss is up-weighted as in the flow-matching objective below. Both masks are training-only, so the runtime path stays calibration-free.

Why the two are complementary. Sight-cone masking acts on attention *topology*: it makes the cross-view graph reflect physical visibility, but does not push traffic along the remaining edges. Tube patch masking acts on input *content*: it removes information recoverable through temporal self-attention, but does not block geometrically nonsensical correlations. Used together at training time, the model may attend across views only where geometry permits and is required to do so to minimize the video objective below (Figure 4).

Video Flow-Matching Objective. The video tower inherits Wan-style flow matching and is trained on event latents before the action tower is attached (Section 5.1). Given clean latents $\mathbf{z}_0^V$, noise ε^V, and a sampled video

timestep $t^V \sim \Phi_V$, we form noisy latents $\mathbf{z}^V_{t^V}$ and regress the flow target $\mathbf{C}^{V\star} = \varepsilon^V - \mathbf{z}^V_0$. All video-side supervision is collected in a single flow-matching loss. By default,

$$\mathcal{L}_V = w_V(t^V) \left\| \hat{\mathbf{C}}^V - \mathbf{C}^{V\star} \right\|^2, \tag{7}$$

with optional per-timestep weighting $w_V(t^V)$, evaluated only on *valid* spatial tokens. Out-of-frame and synthetic border regions are always excluded from this MSE—we refer to this as *border masking* in the loss; it is independent of the sight-cone and tube mechanisms above and is applied in all video runs (Section 5.1). When tube-patch masking is active (p_{tube}>0), denote the masked spatio-temporal tube by $\mathcal{T}$. Tokens inside $\mathcal{T}$ use the same $\mathbf{C}$ target but enter with an additional loss weight λ_{mask}:

$$\mathcal{L}_V \;=\; w_V(t^V) \left(\sum_{u \notin \mathcal{T}} \left\| \hat{C}^V_u - C^{V\star}_u \right\|^2 \;+\; \lambda_{\text{mask}} \sum_{u \in \mathcal{T}} \left\| \hat{C}^V_u - C^{V\star}_u \right\|^2 \right), \tag{8}$$

where $C^{V\star}_u$ is the u-th component of $\mathbf{C}^{V\star}$. Setting λ_{mask}=1 or leaving $\mathcal{T}$=∅ recovers the uniform form of Equation (7). Sight-cone masking affects only cross-view attention and never modifies which tokens contribute to $\mathcal{L}_V$. Our prior-preserving main recipe always keeps border masking and sight-cone attention supervision, disables tube sampling and tube up-weighting (p_{tube}=0 or λ_{mask}=1 with empty $\mathcal{T}$); non-trivial tube settings are reserved for runs that harden cross-view correspondence.

3.3 Event-Centric Action Dynamics Modeling

The action tower is an action DiT with the same depth as the video tower. At each layer, action tokens read multi-view video features from the paired video block; the stack denoises the end-effector trajectory with flow matching. Below we cover layer-wise coupling, video-action temporal alignment, video-action denoising-step mapping, and the action training objective.

Action Transformer and Layer-Wise Coupling. A shared MLP encoder $\mathcal{E}_a : \mathbb{R}^{D_a} \to \mathbb{R}^d$ embeds both the proprioceptive state and the noisy actions into the transformer width; the resulting T_s+T_a tokens (T_s state tokens and T_a action tokens) are processed by a stack of N_{DiT} action blocks, each performing (a) self-attention over the action tokens, (b) a dedicated cross-attention to the state token alone, (c) the cross-attention to the matched video-DiT layer's features, and (d) a gated FFN; an MLP head decodes the action tokens back to $\mathbb{R}^{D_a}$. The state token's separate cross-attention—in addition to participating in the self-attention—keeps absolute proprioception directly reachable from the action tokens at every depth, rather than diluted by the long video key/value sequence. At step (c), action block i cross-attends to video context from the matched video block:

$$\tilde{\mathbf{h}}^V_i \;=\; \text{ViewConcat}\big(\mathbf{h}^V_{\pi(i)}\big) + E_\tau(\boldsymbol{\tau}^V) + E_{\text{abs}}(\boldsymbol{t}_{\text{abs}}), \qquad \pi(i) \text{ indexes the paired video block,} \tag{9}$$

Here $\mathbf{h}^V_{\pi(i)}$ denotes the hidden states leaving video DiT block $\pi(i)$ as N_v separate per-camera token sequences—views not yet merged along the sequence axis. The depth map π pairs action block i with video block $\pi(i)$, so block i cross-attends to features from that video layer; with equal-depth towers we use $\pi(i)$=i. ViewConcat$(\cdot)$ gathers the N_v per-view token sequences at that layer—each spanning all latent frames and spatial patches—and concatenates them along the sequence axis; adding E_τ and E_{abs} yields $\tilde{\mathbf{h}}^V_i$, which serves as the cross-attention keys and values. Coupling runs in one direction only: each action block cross-attends to per-layer video features, and no auxiliary branch is added to the video DiT.

Video-Action Temporal Alignment. Cross-attention requires video keys/values and action queries to share a temporal index. Action self-attention orders the state-action sequence with 1D RoPE; at the video interface we instead add two learnable lookups, E_τ and E_{abs}, to action queries and to the assembled video keys/values. E_τ indexes tokens *inside* the current window; E_{abs} indexes *which* window is active when many fixed-length chunks roll out under one global instruction. We first specify the two window layouts and their positional codes; we then show how the observation-centered layout is instantiated in raw frames, VAE latents, and action tokens without a separate history encoder.

Event-centric window. Used in event-centric pretraining (Section 5.1), where a per-event caption already localizes each segment. We disable E_{abs} and assign every token an integer frame index τ within the window:

$$\tau^V_{(f,h,w)} \;=\; f, \qquad \tau^A_0 \;=\; 0, \qquad \tau^A_{1+k} \;=\; \lfloor k/K_p \rfloor + 1 \quad \text{for } k \in [0, T_a), \tag{10}$$

where each latent frame pools K_p action steps. The state token and zeroth latent frame share τ=0; action tokens partition into T_a/K_p groups aligned with successive future latents; and all spatial tokens in latent frame f share $\tau=f$. The shared lookup $E_\tau(\tau)$, added on both sides of cross-attention, biases each action group toward the matching latent frame.

Observation-centered window. Unified deployment slides fixed-length windows under a global instruction; τ alone cannot distinguish chunks. We extend the window to M history frames, one observation anchor, and N future frames, and activate both embeddings on both sides of cross-attention:

$$\mathbf{h} \mathrel{+}= E_\tau(\tau) + E_{\text{abs}}(t_{\text{abs}}). \tag{11}$$

Here t_{abs} indexes the sliding window; E_{abs} marks which chunk is current, while E_τ spans history indices $-M,\dots,-1$, the anchor at 0, and future indices $+1,\dots,+N$. Section 3.3 summarizes this layout with M=1; below we spell out how one shared codec window materializes it on the video and action sides.

VAE-aligned video stream. The 3D VAE is a 1+4N temporal codec: a keyframe followed by 4N raw frames compresses to 1+N latents under a leading-one/trailing-four rule. In event-centric pretraining the current frame occupies the keyframe slot and the chunk-DiT denoises the N trailing latents. For the observation-centered window we prefix 4M history frames and encode the full 1+4M+4N buffer in one pass; the observation keyframe separates prefixed history from the subsequent 4N raw frames. Under the unchanged codec rule the enlarged window yields 1+M+N latents in temporal order—earliest history to the leading latent, the next four frames to the anchor latent, and each following group of four raw frames to a future latent—with no re-encoding seam between history and future; the chunk-DiT denoises only the N trailing future latents.

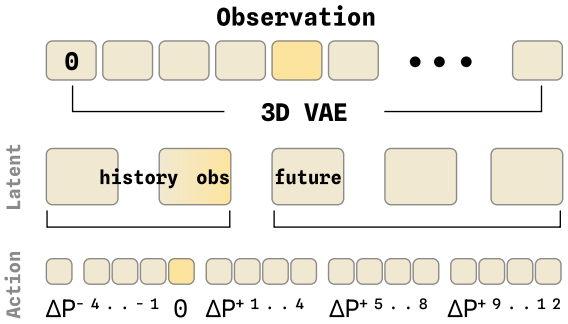


Figure 5 Observation-centered window (M=1). One 3D VAE pass encodes a 1+4M+4N raw buffer into 1+M+N latents. Each latent carries K_p relative-pose action tokens (**0** at the anchor)

Relative-pose action stream. The action side mirrors the 1+M+N latent stream: the tower assigns K_p tokens to each video latent. All non-anchor tokens encode end-effector displacement relative to the observation pose—history tokens back to the anchor, future tokens forward from it; anchor tokens are **0**, fixing the action time origin. Embodiment-absolute proprioception stays in the dedicated state token of the layer-wise coupling paragraph rather than this stream.

Video-Action Timestep Mapping. Under flow matching, video and action follow separate denoising schedules, yet each action block cross-attends to video features from some denoising step; training and inference must therefore specify how action step index j maps to a video step index. We write N_d for the number of action denoising steps on the shared inference schedule (e.g. N_d=50) and a mapping $m(j)$ from each action step to the video step whose cross-attention key/value it reads. Two regimes cover our recipes:

- **Symmetric 1-to-1** (validation): action step j is paired with video step j ($m(j)=j$; equivalently $t^A=t^V$ at each paired step). We use this when both towers are updated end-to-end on small data, denoising the two streams jointly for tight fitting.
- **Asymmetric 1-to-N_d** (default; large-scale training): a moderate-noise *schedule anchor* $s^\star$ is fixed and broadcast to the full action schedule ($m(j)=s^\star$ for all $j\in\{0,\dots,N_d-1\}$). At inference, each denoising pass runs video only through $s^\star$ while action completes all N_d steps; every action step reads cross-attention key/value from that single anchored video forward ($m(j)=s^\star$; Section 5.5).

In our main recipe, the video stream is frozen while the action tower is trained at scale under asymmetric 1-to-N_d mapping. With the video stream frozen, each optimizer step uses only a single anchored video forward as cross-attention evidence while action supervision comes from ground truth, not from a jointly denoised video target. A symmetric 1-to-1 pairing ($t^A=t^V$) is therefore a poor fit: high-noise video features need not match the ground-truth segment, whereas near-clean features retain too little structure to guide control. We pin $s^\star$ at the step that best balances faithful visual structure with usable cross-attention evidence (see Section 5.1) and train

each action forward at an independently sampled noise level $t^A \sim \Phi_A$ while it cross-attends to that same anchor, so the action tower learns across the full action schedule without sampling a mismatched video step. Purely as a training-throughput trick unrelated to inference, we optionally draw $K>1$ independent action noise levels per optimizer step while reusing the same anchored video forward; K amortizes compute and plays no role at rollout. Symbolically,

$$m(j) \;=\; s^\star \;\text{ for all }\; j\in\{0,\dots,N_d-1\}, \qquad t_k^A \overset{\text{u.a.r.}}{\sim} \Phi_A, \;\; k=1,\dots,K, \tag{12}$$

with Φ_A free in support and density; per optimizer step the cost is one video forward plus K action forwards when the throughput trick is enabled (K=1 otherwise).

Action Objective. During action pretraining only the action tower is optimized under flow matching while the video tower remains fixed. Our main runs use v**-prediction**, parallel to the video-side target in Section 3.2. Under the asymmetric 1-to-N_d mapping above, each draw k uses an independently sampled action noise level $t_k^A \sim \Phi_A$ while reusing the same anchored video forward; the primary loss is the mean flow-matching MSE over these draws,

$$\mathcal{L}_A \;=\; \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_A^k, \qquad \mathcal{L}_A^k \;=\; w(t_k^A)\,\big\|\hat{\mathbf{y}}_k^A - \mathbf{y}_k^{A\star}\big\|^2, \tag{13}$$

where $w(t_k^A)$ optionally reweights by noise level as in Section 3.2. By default, $\hat{\mathbf{y}}_k^A$ is the predicted velocity $\hat{\mathbf{v}}_k^A$ and the target is $\mathbf{v}^{A\star} = \varepsilon^A - \mathbf{a}_0$. With optional x**-prediction**, the network outputs clean actions instead ($\hat{\mathbf{y}}_k^A$=$\hat{\mathbf{a}}_0$, $\mathbf{y}_k^{A\star}$=$\mathbf{a}_0$); the weighted MSE above is unchanged. On contact-heavy data (fine-grained dual-arm manipulation, contact-rich insertion, recovery/re-grasp episodes), contact events span only a few frames. v-prediction can weight those frames too lightly, especially at high noise, so we expose x-prediction as an optional mode on such runs. We can also add a Type-II **DCT** auxiliary on the recovered trajectory $\hat{\mathbf{a}}_0^{(k)}$ to emphasize overall motion shape over frame-to-frame jitter (exponential bin decay), down-weighted at high noise by $(1-t_k^A/T)^2$, where T is the maximum action diffusion timestep (the end of the noise schedule):

$$\mathcal{L}_A \;\mathrel{+}=\; \frac{w_{\text{DCT}}}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{DCT}}^k, \qquad \mathcal{L}_{\text{DCT}}^k \;=\; (1-t_k^A/T)^2\sum_{j=0}^{T_a-1} e^{-j/(T_a/4)}\,\big\|\text{DCT}(\hat{\mathbf{a}}_0^{(k)})_j - \text{DCT}(\mathbf{a}_0)_j\big\|^2, \tag{14}$$

with $\hat{\mathbf{a}}_0^{(k)}$ recovered from draw k. Our main runs use v-prediction with w_{DCT}=0, leaving both x-prediction and the DCT auxiliary disabled unless otherwise specified.

3.4 Language-Guided Reasoning

Long-horizon embodied manipulation requires reasoning capabilities beyond static language conditioning, including scene grounding, event decomposition, temporal progress estimation, and rollout-time replanning. Our reasoning module is constructed on top of a Qwen3.5-9B backbone (80, 71) and is accelerated via a compact sequence of continuous latent reasoning states (Staircase decoding).

Given the current multi-view observation $\{\mathbf{V}_0^{(v)}\}_v$ and language instruction ℓ, the VLM produces hidden states $\mathbf{H}_q$. The text tokens are extracted using a boundary mask $\mathbf{m}_{\texttt{txt}}$ and projected into the DiT conditioning space:

$$\mathbf{c}_\ell = \mathcal{M}_T\big(\mathcal{P}_Q(\mathbf{H}_q[\mathbf{m}_{\texttt{txt}}])\big), \tag{15}$$

where $\mathcal{P}_Q$ denotes a learnable projection and $\mathcal{M}_T$ denotes the native text MLP of the DiT backbone. The resulting conditioning sequence is injected into the WAM cross-attention layers together with image tokens.

The standard sequential rollout decoding process suffers from severe computational inefficiency for long-chain latent reasoning. Traditional autoregressive rollout generates latent reasoning states one by one in a serial manner, which demands exhaustive forward propagation of Transformer layers and repetitive calculation of overlapping visual-linguistic features at each reasoning step. This serial dependency bottleneck drastically increases inference latency, restricts the real-time interaction capability of embodied manipulation tasks, and hinders the scalability of long-horizon reasoning rollouts. To address the serial decoding overhead and redundant computation in conventional latent reasoning rollout, we propose a novel Staircase latent decoding paradigm to achieve efficient parallel generation of continuous latent reasoning sequences.

Staircase latent CoT decoding. Instead of autoregressively generating reasoning tokens one-by-one, WALL-WM models reasoning as a compact sequence of K_c continuous latent reasoning states:

$$\hat{y}_{1:K_c} = \{\hat{y}_1, \ldots, \hat{y}_{K_c}\}. \tag{16}$$

The reasoning branch is initialized from the fine-tuned Qwen3.5-9B checkpoint and implemented as a lightweight Mixture-of-Transformers (MoT) structure coupled to the frozen backbone. To enable parallel latent rollout, we partition the Transformer at a relay depth N_r, where lower layers encode shared visual-language grounding features and higher layers progressively specialize into different reasoning steps.

Concretely, only the first latent position traverses the lower Transformer layers, producing a shared relay representation reused across all reasoning positions. The remaining latent states are then generated in parallel through the upper Transformer blocks with independent causal cache updates. This staircase scheduling mechanism transforms the conventional sequential reasoning process into a depth-parallel latent rollout:

$$\hat{y}_{1:K_c} = \mathcal{F}_{\text{stair}}(x; N_r), \tag{17}$$

where N_r controls the relay depth separating shared grounding computation from parallel reasoning computation.

Compared with standard autoregressive decoding, the proposed staircase design avoids repeatedly recomputing low-level visual-language features for every reasoning step, substantially reducing inference latency for long-horizon embodied rollout. The resulting latent reasoning states are fully differentiable and directly injected into the WAM cross-attention pathway without discrete token sampling.

Frozen latent-to-text reconstruction supervision. Rather than directly distilling autoregressive hidden states, WALL-WM supervises latent reasoning through a frozen latent-to-text reconstruction objective (38, 92). The generated latent reasoning states $\hat{y}_{1:K_c}$ are projected by a prefix projector $\mathcal{P}_{\text{pref}}$ into a soft prefix $\mathbf{z}_{1:K_c}=\mathcal{P}_{\text{pref}}(\hat{y}_{1:K_c})$ in the embedding space of a lightweight frozen language model (Qwen3.5-0.8B), which reconstructs the corresponding textual CoT trace autoregressively:

$$P_\phi(r_{1:M_r} \mid \mathbf{z}_{1:K_c}). \tag{18}$$

The training objective is defined as the token-level reconstruction loss:

$$\mathcal{L}_{\text{CoT}} = -\sum_{m=1}^{M_r} \log P_\phi\left(r_m \mid \mathbf{z}_{1:K_c}, r_{<m}\right). \tag{19}$$

Only the staircase reasoning branch and the prefix projector $\mathcal{P}_{\text{pref}}$ are optimized, while the reconstruction language model remains frozen throughout training. Consequently, the latent reasoning states are encouraged to encode compact high-level reasoning semantics rather than replicate exact token-level decoding trajectories. The resulting latent CoT serves as a differentiable, compact, and computationally efficient replacement for explicit textual reasoning during embodied rollout.

Inference modes. The same event-pretrained backbone exposes two complementary inference modes through this language-guided pathway. In **event mode**, the system rolls out in event space: a VLM, human, or agent proposes the next-event description for the current observation, and WALL-WM executes the corresponding variable-length video–action segment before observing the next state, so that rollout follows the natural duration of each task stage rather than a fixed control horizon. In **unified mode**, the staircase decoder emits K_c continuous latent CoT states in a single parallel pass, which slot into the WAM cross-attention exactly where an atomic-instruction condition would path. Because both modes condition the WAM through the same reasoning interface, WALL-WM unifies open-ended, event-driven rollout, and standard fixed-horizon VLA execution within a single model.

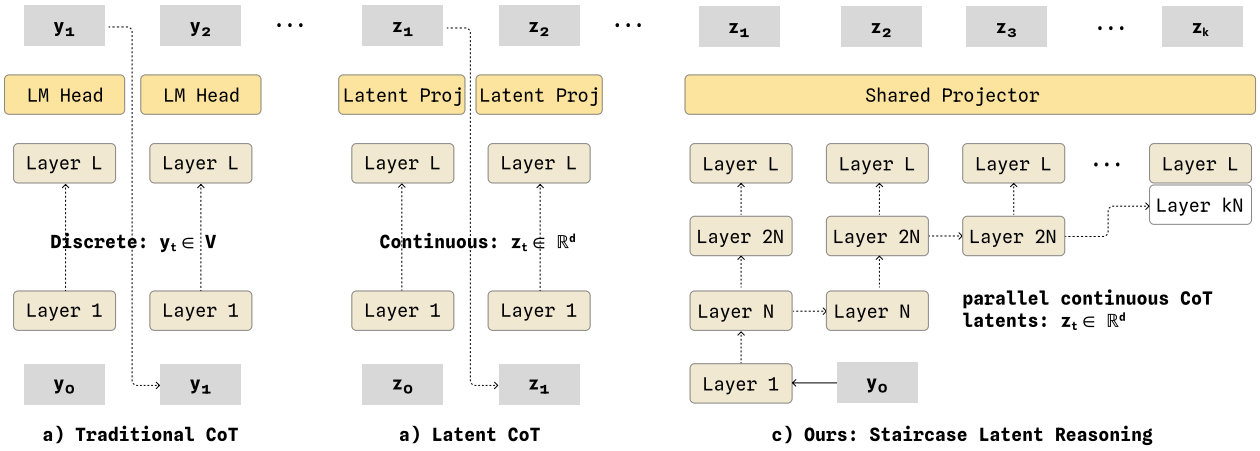


Figure 6 Comparison of three CoT reasoning schedules. *(a)* Traditional CoT autoregressively emits discrete vocabulary tokens. *(b)* Latent CoT replaces discrete tokens with continuous latent vectors while preserving the serial dependency. *(c)* Our staircase latent reasoning relays intermediate hidden states across staggered layer depths, producing parallel continuous CoT latents through a shared projector.

4 Training Data

The WALL-WM data ecosystem contains large-scale internet videos, egocentric videos, robot-free UMI-style recordings, and heterogeneous teleoperation data, with cautious post-processing and rich annotation for general-purpose pretraining. The data ecosystem is organized along five axes: *(i) source coverage*, a data map spanning general internet video, egocentric human video, non-embodiment UMI data, and robot teleoperation/open robot data; *(ii) temporal synchronization and data postprocessing*, a data post-processing framework for training and deployment alignment; *(iii) hierarchical caption granularity*, with every episode annotated at four temporal scales (Task / Subtask / Action / Segment) plus an optional human-annotated layer; *(iv) joint vision-language clustering paired with action clustering*, used to balance the dataloader and to surface long-tail non-nominal trajectories; and *(v) targeted recovery augmentation*, covering contact-rich action regions and geometry / occlusion regimes that are difficult to capture through real-robot data collection.

4.1 Data Composition

We organize the WALL-WM dataset as a **data-source map** rather than a strict hierarchy. The map separates source families by viewpoint and action availability. General internet video provides broad visual and temporal-dynamics priors at a scale that no embodied corpus can match; in our current build this includes a 1.2M-clip OpenVID (55) slice together with other web video sources.

Egocentric and human-action videos narrow this prior toward first-person manipulation geometry without requiring robot actions. Robot-free UMI-style recordings add deployment-like hand and camera geometry with retargetable controller trajectories, while heterogeneous robot datasets and self-collected teleoperation provide real `[video+action]` pairs across platforms. Human-intervention and failure-recovery data are treated as a central source because they can be sampled from or attached to multiple quadrants whenever contact-rich correction behavior is needed.

The robot-data quadrant combines embodied public datasets and geometry-distribution-consistent self-collected data, where both observation geometry and action measurement are aligned with the deployment platform. Our internal self-developed embodiment suite spans four deployment platforms: high-performance desktop bimanual robot arms; two mobile robot platforms,

Figure 7 Internal self-developed deployment platform.

Figure 8 Data-source map of the WALL-WM dataset. The four quadrants group general internet video, egocentric video, non-embodiment data, and heterogeneous teleoperation/open robot data; the center denotes human-intervention and failure-recovery data used to enrich contact-rich corrections. Data sources include OpenVID (55), HD-VILA (79), Ego4D (25), EPIC-KITCHENS (18), DROID (40), AgiBot World (10), and self-collected data (73).

QUANTA X1 and QUANTA X1 Pro; and the wheeled humanoid
robot QUANTA X2, which is equipped with high-degree-of-freedom dexterous hands. Recovery and takeover episodes explicitly cover the action solution space near contact-rich events and provide failure-recovery signal that nominal demonstrations rarely supply by sheer volume.

Non-Embodiment Collection with a Wearable UMI-Style Rig. The non-embodiment quadrant of Figure 8 is collected with a wearable, robot-free rig rather than a teleop'd robot. An operator wears a VR-tracked headset carrying multi-view egocentric cameras and a pair of handheld grippers whose physical geometry is calibrated to match the deployment robot's end-effectors (Figure 10, left); the resulting 6-DoF controller trajectories, after IK retargeting against the deployment robot's URDF, yield an admissible action stream, while the ego+dual-wrist video already sits in the geometry the real robot will see. We use our own **XRZero-G0** system (73) for this quadrant (Figure 10, right): it sits alongside master-slave and VR teleoperation rather than replacing them, and contributes the VR-tracked acquisition stack with millimeter-level pose accuracy. Because the operator is no longer attached to a robot, collection throughput is no longer bound by robot time; in the production recipe a small real-robot *anchor* fraction is paired with a much larger volume of no-embodiment clips, following the few-shot physical-anchoring regime characterized in (73).

Collection Protocols: Structured and Unstructured Streams. The self-collected robot data in Figure 8 is drawn from two collection protocols whose distinction sits entirely on the collection side, not on the training side. *Structured* collection follows a predefined task scope and reset protocol: each episode is one teleop'd demonstration of a named task, with explicit start / goal conditions and clean episode boundaries. *Unstructured* collection, by contrast, lets operators move the robot freely in the deployment scene without committing to a task scope, a reset protocol, or an episode boundary in advance, yielding a long, multi-event in-distribution motion stream that a demonstration-only dataloader would have to discard. Both protocols feed the *same* downstream pipeline: the hierarchical caption schema of Section 4.3 segments the raw stream into Task / Subtask / Action / Segment spans *after* collection, and the clustering pass of Section 4.4 treats the segmented clips identically regardless of which protocol produced them. Admitting unstructured collection is what lets this source scale beyond the natural ceiling of teleop'd demonstrations: it removes the per-episode protocol overhead that otherwise binds collection throughput, while the caption-then-cluster pipeline absorbs the resulting heterogeneity into the same balanced sampler used everywhere else. The overall distribution statistics of `[video+action]` data can be found in Figure 9.

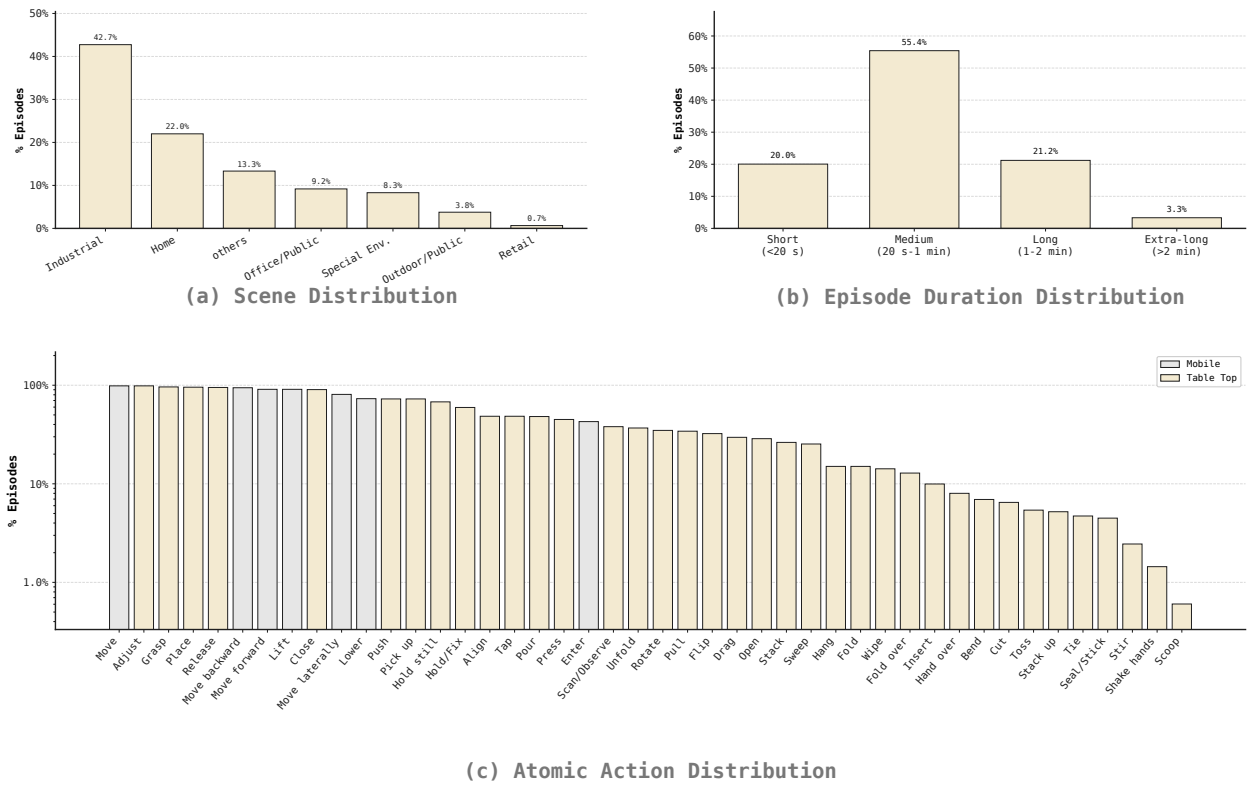


Figure 9 Distribution statistics of the [video+action] data. The plots show normalized coverage across scene categories, episode-duration ranges, and atomic-action labels. Scene and duration categories are reported as percentage distributions, while atomic actions indicate the percentage of episodes containing each action label.

Figure 10 XRZero-G0 no-embodiment collection rig and its place alongside teleoperation (73). *(Left)* The wearable rig from two views: a VR-tracked headset carries three egocentric cameras (one top + two wrist-side feeds shown), an operator-borne backpack handles synchronization and edge-side compute, and a pair of handheld grippers reproduces the deployment robot's end-effector geometry; the inset thumbnails show the three on-rig views (left wrist / top / right wrist) that the policy will eventually consume. *(Right)* XRZero-G0 sits alongside master-slave and VR teleoperation rather than replacing them: in the production recipe a small real-robot *anchor* fraction (the two teleoperation modes) is paired with a much larger no-embodiment volume (the XRZero-G0 rig), realizing the few-shot physical-anchoring regime characterized in (73).

4.2 Deployment-Aligned Temporal Synchronization and Data Postprocessing

For the non-embodiment data and robot data collected from tele-operation, the [video+action] layers are only useful if the visual observation and the action stream refer to the same physical instant. However, this alignment is not guaranteed in practice: camera encoding, controller logging, teleoperation middleware, and disk writing can introduce a nearly constant video–action offset within a recording source. If left uncorrected,

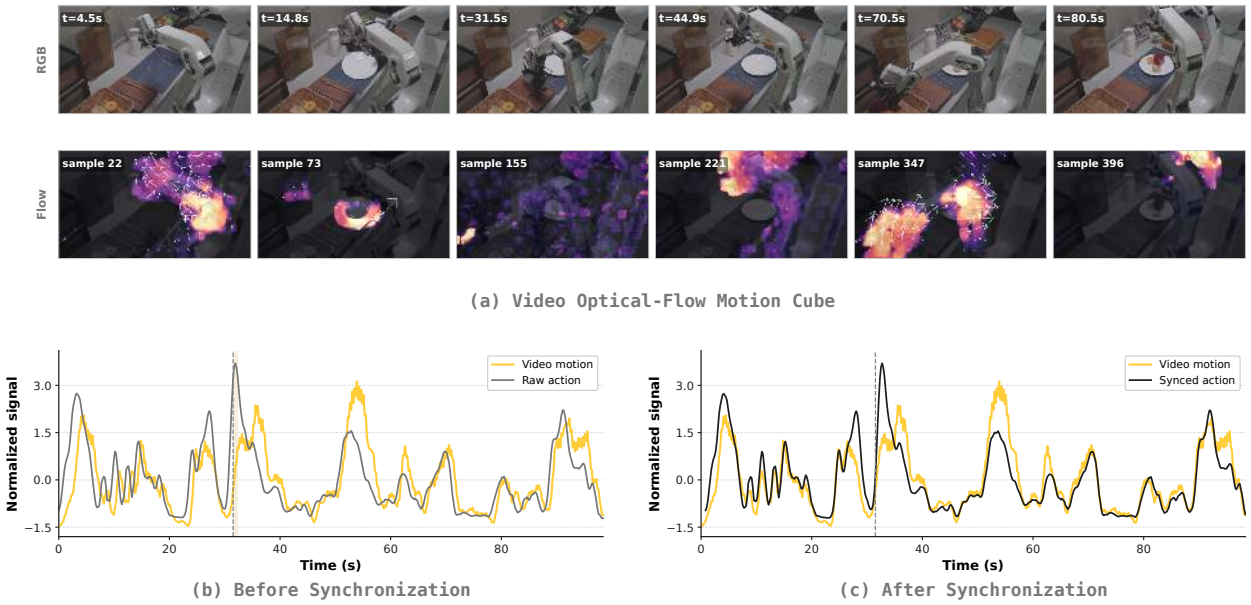


Figure 11 Temporal Synchronization of `[video+action]` layers. We illustrate the process of temporal synchronization for a specific timestep. (a) Calculating the optical-flow motion cube from video data: we grayscale and normalize the camera RGB images, then calculate the *optical flow* from the video data. (b) Before synchronization: there is a constant lagging between video signal and action signal. (c) We performed temporal synchronization calculating the correlation between the video signal and the action signal, and synchronized the action signal based on the results.

the model is trained to associate an image at time t with an action from $t + \Delta$ or $t - \Delta$. This is especially damaging near contact, where a few frames can change the semantic state from "approaching" to "touching", "grasping", or "recovering". It can also lead to undesirable shaking behavior when the trained policy is deployed on robot hardware.

Therefore, we conduct **temporal-synchronization** of the `[video+action]` layers before captioning, clustering, and sampling. Specifically, for each episode, we construct a visual motion signal by measuring frame-to-frame image change across available ego and wrist cameras, and an action-motion signal by taking the finite difference of the left and right end-effector position streams. Both signals are smoothed and normalized, and we sweep a small integer lag window to find the offset that maximizes their correlation. The estimate is aggregated over cameras and action channels so that a transient occlusion or a stationary gripper does not dominate the decision. The illustration of this whole process can be found in Figure 11.

Rather than assuming perfect video-action timing a priori, this post-processing stage explicitly estimates the residual phase difference and calibrates it before downstream use. For example, at 20 FPS, if an episode is estimated to have a two-frame offset, corresponding to roughly 100 ms, the synchronizer applies the corresponding temporal shift to align the logged action stream with the encoded video timeline. The correction is applied deterministically: the action sequence is re-indexed to the visual frame timeline, boundary frames introduced by the shift are trimmed, and continuous action fields are resampled only after the offset has been removed. Episodes whose best-lag score is weak, whose source-level vote is inconsistent, or whose required correction falls outside the expected window are not silently used; they are either quarantined for manual inspection or down-weighted until a source-level correction can be verified.

After synchronization, we run a second cleaning pass for **deployment alignment**. This pass removes episodes with missing camera streams, non-monotonic or duplicated action records, abnormal FPS, large frame/action length mismatch, invalid robot states, kinematic discontinuities, gripper-state corruption, or motion segments that violate the deployment robot's reachable workspace after retargeting. For structured demonstrations, failed resets and truncated starts/goals are removed. For unstructured streams, long recordings are split only after synchronization, so that discovered Task/Subtask/Action/Segment spans inherit a consistent video-action clock. The cleaned output of this stage is a canonical episode format: synchronized multi-view video, aligned action

records, calibrated camera metadata, robot-state metadata, source identity, confidence flags, and cleaning provenance. These flags are consumed downstream by the hierarchical captioner and by the cluster-balanced dataloader.

4.3 Event-Centric Hierarchical Captioning

Every episode in the action-paired sources of Figure 8 is annotated with a temporally nested caption hierarchy whose spans are *temporally grounded* on atomic manipulation actions (66)—short, executable primitives such as reach, grasp, close, lift, and place—rather than on fixed frame windows. Concretely, we segment the synchronized video–action stream at action boundaries first, then assign captions so that each linguistic interval aligns with the same physical interval in vision and control. Rather than assigning a single free-form description to an entire trajectory, we decompose each episode into four core levels of temporal abstraction, together with an optional human-annotated layer. This hierarchy is designed to preserve both the global intent of the demonstration and the local event structure that determines how the robot actually completes the task.

The four core levels are *Task*, *Subtask*, *Action*, and *Segment*. The *Task* level is an episode-global string that summarizes the overall objective, such as opening a drawer, placing an object into a container, or wiping a surface. The *Subtask* level partitions the episode into a small number of contiguous, semantically meaningful stages. These stages typically correspond to intermediate goals, for example approaching the target object, establishing a grasp, transporting the object, and placing or releasing it. The *Action* level further refines each subtask into shorter manipulation primitives, such as reaching, aligning the gripper, closing the fingers, lifting, translating, inserting, or retracting. Finally, the *Segment* level provides the finest temporal decomposition, capturing short and localized events that may last only a small fraction of the full trajectory. A fifth *Human* level stores manually annotated segments. This layer is populated only for the subset of episodes that received human annotation, and serves two complementary purposes. First, it provides high-quality supervision for validating the automatically generated hierarchy. Second, it gives us a trusted reference for evaluating caption accuracy, temporal boundary quality, and the consistency of semantic labels across tasks. In practice, the human layer is not required for every episode; the core four-level hierarchy allows the dataset to scale, while the human subset anchors the annotation process and supports quality control.

Figure 12 Four-level caption tracks for a single episode. The visualization stacks the four core lanes shown in the figure: Task (L3), Subtask (L2), Action (L1), and Segment (L0), aligned to the video timeline.

This multi-level schema is especially important for robotic recovery behavior. Many useful demonstrations are not perfectly linear executions of a task. They contain regrasps, failed contact attempts, small pose corrections, retries after slipping, and other short recovery motions. These events are often essential for robust policy learning, because they expose the model to states that arise when execution deviates from the nominal path. However, if an entire episode is represented only by a single caption, these corrective behaviors are averaged into the global task description and become difficult for the model to identify. The trajectory may still be labeled as “pick up the object,” even though a crucial portion of the episode involved missing the grasp, repositioning the wrist, and trying again. The hierarchical caption structure makes these events explicit. At the task level, the

episode remains associated with its intended goal. At coarser subtask and action levels, the model observes the main phases of execution. At the finest segment level, short corrective behaviors can be localized in time and assigned their own descriptions. This enables the dataloader to sample or re-weight specific temporal regions, rather than treating all frames within a successful episode as equally informative. As a result, recovery-relevant intervals can be emphasized during training without discarding the broader task context.

4.4 Cluster-Balanced Sampling over Vision-Language and Action

The production recipe trains on only a small subset sampled from our large-scale self-collected embodied corpus, selected by the cluster-balanced sampler described below. This subsampling is not merely a budget concession: it follows from the observation that the raw embodied corpus is highly long-tailed, while the multi-level caption schema of Section 4.3 exposes fine-grained statistics over vision, language, and action. In particular, the caption windows are segmented according to action boundaries before clustering, rather than by uniform temporal slicing. This action-aligned decomposition turns each long demonstration into semantically localized units, making the resulting language and vision–language distributions substantially more balanced than the original task-level mixture.

On top of this multi-level schema, WALL-WM runs two offline clustering passes whose purpose is to give the dataloader a controllable sampling distribution. The first is *vision–language* (VL) clustering: a frozen multimodal encoder maps each visual observation and caption pair into a joint embedding, and a clustering procedure partitions this space into topic clusters that summarize the corpus's instruction-scene coverage. The second is *action* clustering: action chunks are clustered separately in trajectory space, where the long tail concentrates non-nominal motion such as recoveries, re-grasps, retries, and contact-driven corrections. These are precisely the behaviors that the action-aligned multi-level schema is designed to surface. Importantly, after action-based decomposition from task-level captions to finer action and segment captions, the language-only and vision–language clustering views become more evenly distributed, with fewer samples absorbed by a small number of dominant topics. This provides a practical basis for balanced sampling: rare-but-important instruction-scene and trajectory modes become explicit sampling units instead of being overwhelmed by frequent verbs and nominal motions.

During training, the dataloader samples batches that balance both VL clusters and action clusters. VL balancing improves coverage over instruction-scene combinations, while action balancing repeatedly exposes the action head to trajectories that share common verbs but require different motions. This is the regime in which a vanilla SFT objective on the raw long-tailed mixture would tend to collapse toward dominant targets. Both clusterings are precomputed offline; training consumes only cluster assignments, so cluster-balanced sampling adds neither encoder forward passes nor optimizer state to the existing recipe.

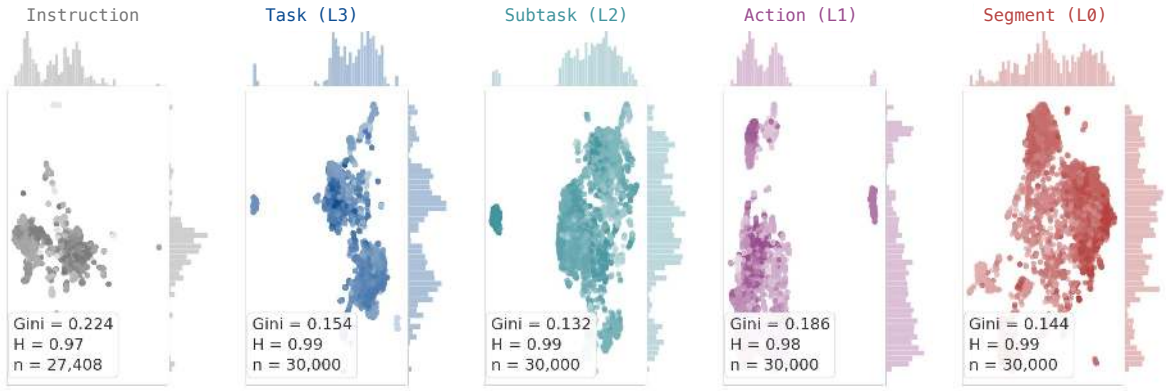


Figure 13 Caption-granularity distributions for balanced sampling. The five panels compare 30K samples at Instruction, Task (L3), Subtask (L2), Action (L1), and Segment (L0) granularities after action-boundary segmentation. Finer caption levels expose more localized long-tail modes that are consumed by the cluster-balanced dataloader.

4.5 Recovery Data via Contact-Rich Random Initialization

Pretraining is augmented with *recovery / takeover* episodes that explicitly cover the action solution space around contact-rich events. For each such event e, we define a local contact-pose distribution $p(\mathbf{q} \mid e)$ over end-effector configurations $\mathbf{q} \in \mathbb{R}^{D_q}$. Its support lies in a small geodesic ball $\mathcal{B}_\epsilon(\mathbf{q}_e^\star)$ centered at the nominal contact pose $\mathbf{q}_e^\star$, so the sampled states remain local to the event while preserving its contact geometry.

The resulting pretraining distribution is

$$\tilde{p}_{\text{train}}(\mathbf{q}, e) = (1 - \alpha)\, p_{\text{nominal}}(\mathbf{q}, e) + \alpha\, \mathbb{E}_e[p(\mathbf{q} \mid e)]\,,$$

where α is a small mixing weight. In practice, we sample the second term by perturbing the robot initialization around $\mathbf{q}_e^\star$ and then either replaying the original demonstration from the perturbed state or collecting a fresh recovery rollout. This creates controlled local coverage around each contact event, rather than observing only the single nominal trajectory that passes through it.

This recovery mixture occupies the central recovery source in Figure 8 and complements the cluster-balanced sampling protocol of Section 4.4. Clustering surfaces non-nominal trajectories that already exist in the corpus, while recovery initialization actively creates such trajectories in contact-space regions where the nominal data is too sparse for clustering alone to provide adequate coverage.

5 Training and Inference Recipe

This section details the full training and inference recipe for WALL-WM. Table 1 summarizes which components are trained at each stage. We first pretrain the layer-coupled video-action denoiser under the *next-event prediction* scheme of Fig. 2(A) (Section 5.1), training the video tower first and then the action tower with the video tower frozen. We then adapt the VLM text conditioner (Section 5.2) and train the Staircase decoder for unified-mode parallel chain-of-thought (Section 5.3). An optional fine-tuning stage adapts both towers to the observation-centered history window under the *next-chunk prediction* scheme of Fig. 2(B) (Section 5.4); event-mode rollout requires only the preceding stages. Section 5.5 concludes with the event-mode and unified-mode inference protocols supported by the assembled stack.

Table 1 Trainable / frozen matrix. ✓ indicates a component trained in that stage; ✗ indicates a component kept frozen; "–" indicates a component not yet attached. Columns follow the training pipeline in order: event-centric video and action pretraining (Section 5.1, first two columns), VLM text-conditioning (Section 5.2), Staircase distillation (Section 5.3), and optional next-chunk adaptation (Section 5.4). Video and action pretraining use frozen T5 features; next-chunk adaptation keeps T5 frozen for global instructions while updating the V-A denoiser. The VLM stack attaches from stage 3 onward; only MoT branches train in the Staircase stage.

Component	Training stages				
	Video PT	Action PT	VLM text	Staircase	Next-chunk
3D causal VAE $\mathcal{E}_V$	✗	✗	✗	✗	✗
T5 text encoder	✗	✗	✗	–	✗
VLM backbone (Qwen3.5-9B)	–	–	✗	✗	✗
VLM project-out / auxiliary heads	–	–	✓	✗	✗
Video DiT (incl. view attention)	✓	✗	✗	✗	✓
Action DiT (incl. layer coupling)	–	✓	✗	✗	✓
Staircase MoT branches	–	–	–	✓	✗

5.1 Event-Centric Video-Action Pretraining

Event-centric video-action pretraining implements *next-event prediction* on individual semantic events, each providing paired ground-truth video and action for one manipulation segment. Training proceeds in two sequential stages: video pretraining, followed by action pretraining. In the first stage, we train only the video DiT on event latents under the Wan-style v-prediction flow-matching objective $\mathcal{L}_V$ of Equation (7), adapting the inherited Wan prior into an embodied, multi-view event-future predictor; the action tower is not yet attached.

In the second stage, we freeze the video DiT and train only the action tower under action flow matching to predict the aligned end-effector trajectory.

Video Pretraining. Video pretraining includes the cross-view branch of Section 3.2, whose output projector is initialized to zero. Starting from the Wan prior, the DiT is conditioned on the current multi-view observation (one keyframe per camera) and on event-aligned captions from Section 4.3, encoded by a frozen T5; video timesteps are sampled uniformly. Because event spans vary in duration, we truncate each clip to at most 65 latent frames (corresponding to 129 raw frames under stride-2 subsampling). We further apply a length-aware caption-drop schedule: let L_e denote the raw event span length in frames and let $\rho(L_e) \in [\rho_{\min}, \rho_{\max}]$ be the probability of omitting the event caption on a training step,

$$\rho(L_e) = \begin{cases} \rho_{\min}, & L_e \le L_{\min}, \\ \rho_{\min} + (\rho_{\max} - \rho_{\min}) \dfrac{1 - \cos\!\left(\pi \frac{L_e - L_{\min}}{L_{\max} - L_{\min}}\right)}{2}, & L_{\min} < L_e < L_{\max}, \\ \rho_{\max}, & L_e \ge L_{\max}, \end{cases} \tag{20}$$

We set $\rho_{\min}$=0.1, $\rho_{\max}$=0.9, $L_{\min}$=129, and $L_{\max}$=220, with $L_{\min}$ matching the truncation cap above. On caption-free steps, training reduces to *observation-anchored* future synthesis: conditioned only on the current multi-view observation, the DiT must infer a physically plausible continuation of contact and end-effector dynamics rather than lexically specified sub-goals. We prune quasi-static frames during event construction so that flow-matching supervision concentrates on segments with salient end-effector motion. We additionally use three training practices: (i) border masking that excludes out-of-frame regions and synthetic black borders from $\mathcal{L}_V$ (Equation (7)); (ii) an EMA of the DiT weights (decay β_{ema}=0.9999) updated each step; and (iii) resolution/length bucketing that groups events of similar latent shapes to amortize padding across variable-length clips. Training draws from the cluster-balanced sampler of Section 4.4.

Action Pretraining. After video pretraining, we load the resulting video DiT and keep it fixed. We then optimize only the action transformer under action flow matching on the same event-centric video-action pairs of Section 4.3, disabling the video flow-matching loss so that action gradients do not update the video tower. Following the asymmetric 1-to-N_d mapping of Section 3.3, we pin the frozen video forward to the schedule anchor $s^\star$=45 on a 50-step denoising schedule (selected empirically, with a small jitter window during training). This single anchored video forward supplies cross-attention keys/values to every action denoising step ($m(j)$=$s^\star$ for all $j \in \{0, \dots, N_d - 1\}$, with N_d=50). We additionally set K=6 parallel action-noise draws per optimizer step to reuse the same anchored video forward; this throughput trick is training-only and is not used at inference. The action tower is randomly initialized; training follows the event-centric layout of Section 3.3 and uses the cluster-balanced sampler of Section 4.4.

5.2 VLM Text-Conditioner Pretraining

In this stage we adapt only its project-out head and two lightweight auxiliary heads, without modifying the downstream DiT. A single VLM forward pass over the current event is supervised to emit two signals: (i) a *next-event* description—a task-decomposition caption of the upcoming sub-task—and (ii) an estimate of the *remaining time* until the current event terminates. The hidden states from this same forward pass provide the conditioning features for the DiT, so the latency profile matches a standard T5 conditioning path.

Crucially, the VLM hidden states are aligned to the downstream DiT's **T5** feature space, making the upgraded VLM a *drop-in replacement* for T5 from the DiT's perspective. This alignment preserves the DiT-side text-conditioning prior (the DiT continues to consume text in the geometry it was trained on) while importing the broader reasoning bandwidth of a multimodal VLM, including scene-grounded disambiguation, task decomposition, and a temporal anchor via remaining-time prediction. Throughout this stage, the VLM backbone remains fully frozen; only the project-out head that produces $\mathbf{c}_\ell$, the next-event head, and the remaining-time regressor are trained. Concretely, we minimize a three-term objective that aligns the VLM conditioning $\mathbf{c}_\ell^{\text{VLM}}$ (the VLM-produced version of $\mathbf{c}_\ell$ from Section 3.4) to the corresponding $\mathbf{c}_\ell^{\text{T5}}$ produced by the original T5 encoder on the same instruction, while supervising the two auxiliary heads,

$$\mathcal{L}_{\text{text}} = \lambda_{\text{align}} \left\| \mathbf{c}_\ell^{\text{VLM}} - \mathbf{c}_\ell^{\text{T5}} \right\|^2 + \lambda_{\text{next}} \, \text{CE}\!\left(\hat{c}_{\text{next}}, c_{\text{next}}\right) + \lambda_{\text{time}} \, \text{Huber}\!\left(\widehat{\Delta t}, \Delta t\right), \tag{21}$$

where c_{next} is the ground-truth next-event caption (a discrete token sequence, written with the same c form as the per-event caption c_e of Section 3.3) and $\hat{c}_{\text{next}}$ is its prediction; the first term aligns features to the original T5 encoding, the second is a token-level language-modeling loss on the next-event caption, and the third is a Huber regression on the remaining-time scalar Δt. The three loss weights $\lambda_{\text{align}}, \lambda_{\text{next}}, \lambda_{\text{time}}$ are kept fixed throughout this stage.

5.3 Staircase Distillation

The staircase latent CoT module in Section 3.4 is implemented as a lightweight reasoning branch coupled to the frozen Qwen3.5-9B backbone through a Mixture-of-Transformers (MoT) architecture. Initialized from pretrained Qwen3.5 weights, the reasoning branch inherits the semantic reasoning capability of the original autoregressive model while enabling efficient parallel latent rollout. Given the visual-language input, the staircase module generates a compact sequence of continuous latent reasoning states $\hat{y}_{1:K_c}$, which serve as implicit reasoning variables instead of explicit textual tokens. These latent states are directly injected into the WAM cross-attention layers during embodied rollout.

For supervision, the latent sequence is projected by a prefix projector $\mathcal{P}_{\text{pref}}$ into a soft prefix representation $\mathbf{z}_{1:K_c}$ in the embedding space of a frozen lightweight Qwen3.5-0.8B language model, which autoregressively reconstructs the corresponding textual CoT trace $r_{1:M_r}$:

$$P_\phi(r_{1:M_r} \mid \mathbf{z}_{1:K_c}). \tag{22}$$

Training optimizes the latent-to-text reconstruction objective $\mathcal{L}_{\text{CoT}}$, with gradients applied only to the staircase reasoning branch and the prefix projector $\mathcal{P}_{\text{pref}}$. All other components remain frozen during this stage.

5.4 Next-Chunk Adaptation

Following event-centric pretraining (Section 5.1), this optional stage fine-tunes the event-pretrained backbone under *next-chunk prediction* on the observation-centered layout of Section 3.3. Under one global instruction, history frames localize each H_a-step chunk—H_a denoting the fixed number of action steps per chunk, the fixed-length counterpart of the variable event horizon used in event-centric pretraining—substituting for the per-event captions used during event-centric pretraining.

Observation-centered fine-tuning. Global instructions come from the Task level of Section 4.3 and are encoded by the same frozen T5 as in pretraining; window geometry, relative-pose action targets, and the extended E_{abs}/E_τ indices follow Section 3.3. Both DiT towers are updated on these fixed-shape windows under the same asymmetric 1-to-N_d anchor protocol ($s^\star$=45, $m(j)$=$s^\star$, with optional training-time K=6 action-noise reuse during optimization).

Cluster-balanced sampling. Sliding unified windows shift the instruction-trajectory co-occurrence relative to event-centric pretraining, so we rerun the offline VL and action clustering passes of Section 4.4 on history-conditioned windows from the same corpus before optimization. The dataloader balances the resulting clusters to retain coverage over rare global instructions and non-nominal trajectories surfaced by Sections 4.3 and 4.5, without adding any encoder forwards or optimizer state at train time.

5.5 Inference Recipe

The assembled stack supports two inference modes that mirror the two schemes of Fig. 2. Event-mode inference runs *next-event prediction* on the event-centric scheme (A) layout and needs neither history nor a fixed chunk contract. Unified-mode inference runs *next-chunk prediction* on the observation-centered scheme (B) layout after Section 5.4, and is used mainly for conventional fixed-horizon VLA evaluation. Both modes use the asymmetric 1-to-N_d anchor protocol of Section 3.3 ($s^\star$=45, $m(j)$=$s^\star$ on the default 50-step schedule).

5.5.1 Event-Mode Inference

At each rollout step the video and action DiTs condition on the current multi-view observation, proprioceptive state, and a *next-event description* for the upcoming semantic event. The description comes from a human or

from the fine-tuned Qwen3.5-9B heads of Section 3.4, which also emit a remaining-time estimate for the present event. The model denoises the full event-aligned video and action segment in the scheme (A) variable-length window; when execution completes, the observation advances and a new next-event description conditions the next segment.

5.5.2 Unified-Mode Inference

Unified-mode inference rolls out fixed H_a-step video-action chunks on the scheme (B) layout, conditioning at each step on the current observation, a configurable history window, and a global instruction under the same anchor protocol. During unified rollout, three interchangeable sources supply the per-chunk text-side context $\mathbf{c}_\ell$, either by reusing one encoded global instruction or by providing a fresh caption each chunk, and may be mixed without retraining.

- **Gradient-continuous source.** The text encoder of Section 3.4 maps the global instruction once into a continuous $\mathbf{c}_\ell$ that is fed verbatim into cross-attention and reused across all rolling chunks, with no discrete CoT bottleneck between language and the denoiser.
- **Atomic-instruction source.** An upstream planner or human operator provides one short instruction for each fixed H_a-step chunk boundary rather than for a variable-length semantic event, and the same text encoder maps each instruction to the $\mathbf{c}_\ell$ that drives the corresponding chunk.
- **VLM-CoT source.** The Staircase decoder of Section 3.4, together with the cloned MoT branches trained in Section 5.3, emits CoT latents in a single parallel forward, and their representations serve as the per-chunk text-side context for end-to-end VLA rollouts.

Because every source only replaces the cross-attention text input, sources can be switched mid-rollout without altering denoiser state.

6 Infrastructure

6.1 General Frameworks

Training and serving our model at scale exposes a set of systems challenges that differ in kind from those of pure language models. The co-existence of a heterogeneous backbone , pretrained latent prediction expert and randomly initialized action expert, and a mixed objective of video generation and action prediction, and tight latency budgets at deployment together create bottlenecks that off-the-shelf training and inference stacks do not address. We therefore develop a series of infrastructure-level optimizations tailored to our model.

Muon Optimizer. We adopt Muon as the optimizer for the majority of modules in our model, motivated by its consistent gains in convergence speed and training stability. However, the Newton-Schulz (NS) iteration introduces significant overhead when applied directly to sharded training, in the vanilla implementation, the optimizer step alone can approach the 2x cost of the forward and backward passes combined. To make Muon practical at scale, we implement **DMuon**, a distributed realization of Muon designed for hybrid parallelism and decoupled from the underlying training framework. DMuon resolves the granularity mismatch between matrix-level NS and sharded parameters through co-designed distributed execution, reducing the optimizer step to a secondary cost rather than the dominant bottleneck.

- **Pipeline Scheduling.** We adopt a dedicated-ownership scheme in which a constrained Longest-Processing-Time assignment maps each matrix parameter to a unique owner rank. The native all-reduce / reduce-scatter pattern is replaced by reduce / broadcast, with the post-step broadcast issued asynchronously on a dedicated CUDA stream and overlapped with the subsequent forward pass, while an adaptive runtime monitor falls back to synchronous broadcast under network contention. With the fine-grained scheduling optimization, both the computation and communication redundancy are eliminated.
- **Kernel Optimization.** The Gramian formulation surfaces a symmetric structure that general-purpose GEMM cannot exploit, leaving nearly half of the tile-level computation redundant by construction. We reclaim it through a CuteDSL kernel aligned to the symmetric factor, with shape-aware autotuning sustaining the speedup across the matrix geometries encountered in training.

Kernel Library. We develop a customized kernel library spanning both training and inference, motivated by the observation that stock kernels operate at single-operator granularity and consequently incur unnecessary spills to global memory on tensors that are produced and immediately consumed within a small neighborhood of the dataflow graph. We identify a set of recurring fusable patterns whose arithmetic intensity is bounded by memory bandwidth rather than tensor-core throughput in their default implementations. We consolidate these patterns into composite kernels that keep intermediate activations in registers or shared memory across the fused region, shifting the effective roofline operating point toward the compute-bound regime and sustaining high GPU utilization across the dominant cost centers of the model.

Our kernel library is built on top of TVM FFI, a language-agnostic foreign function interface that exposes compiled kernels through a stable C ABI with zero-copy tensor handles, bypassing the per-call cost of the PyTorch dispatcher and the GIL while remaining interoperable with the PyTorch training and inference frontend. On top of this dispatch layer, we provide a complete wrapper that mirrors the native PyTorch op interface, so that kernels from our library can be substituted for their PyTorch counterparts without any change to the surrounding model code. This design preserves the host-side launch savings while making it straightforward to maintain numerical consistency between training and inference, since both paths invoke the same compiled kernel through the same call surface.

Fine-Grained Overlapping. We introduce a view attention alongside the original spatial-temporal attention, resulting in two sequential attentions per layer. Under Ulysses sequence parallelism, each attention requires a pair of all-to-alls to redistribute the sequence between the sequence and head dimensions, so a naive implementation serializes four all-to-alls per layer on the critical path and substantially inflates the communication cost of adding view attention. To mitigate this, we apply fine-grained scheduling to hide the back-to-back communication within attention computation, eliminating most of the added communication overhead.

Multi-Event Sequence Packing. A common training recipe first *offline*-encodes and caches video latents at *entire-episode* granularity, then samples from those caches during optimization. Whenever a minibatch is formed, the pipeline must therefore load and re-materialize a full episode's worth of latents (or repeat the VAE encode on the fly); because episode lengths vary widely, episodes cannot be packed into uniform tensors without aggressive padding or truncation, and the effective batch size often collapses in practice. We instead pack *multiple events* into a single long sequence at the dataloader level and train on the packed sequence in parallel, with an attention mask that blocks cross-event leakage. Events are concatenated up to a fixed total length, so each training step always runs at the full configured effective batch size and the GPU stays close to compute-bound; the per-step cost is amortized across all packed events instead of being paid once per episode.

6.2 Model Compression

World models are notoriously expensive to deploy, the underlying diffusion-style backbone requires tens of denoising steps per generated frame, and each step itself involves a full forward pass through a multi-billion-parameter network, making naive deployment fundamentally incompatible with the latency budgets of real-time robotic control. To close this gap, we apply two complementary compression techniques along orthogonal axes—distillation to reduce the number of denoising steps, and FP8 quantization to lower the per-step compute and memory cost.

Distillation. To reduce the number of denoising steps, we adopt *distribution-matching distillation* (DMD) (84, 85), which trains a few-step student generator to align its output distribution with that of the multi-step teacher rather than regressing onto teacher trajectories pointwise. Concretely, the distribution-matching term supervises the student via the gap between two score estimates—one from the frozen teacher and one from an auxiliary *fake score network* trained online to track the student's evolving distribution—pushing the student toward regions where its own samples are under-represented relative to the teacher. We further adopt a *joint distillation* objective that retains the original action-prediction loss alongside this distributional term, so that the few-step student is simultaneously aligned with the teacher's video distribution and anchored to the action supervision it was pre-trained on—preventing the action head from drifting as the denoising trajectory is compressed. Ablating the latter degrades action MAE by 53%, confirming that compressing the denoising trajectory under distributional supervision alone causes the action head to drift away from its pre-trained calibration. We initialize the student from teacher weights and distill it down to a small number of sampling

steps; at deployment only the student is retained, yielding an order-of-magnitude reduction in denoising steps with acceptable generation quality.

FP8 Quantization. The second axis of compression targets per-step compute and memory cost rather than step count, we adopt numerical quantization to the dominant matrix multiplications of the distilled student into 8-bit floating-point precision. FP8 is uniquely well-suited to this setting—its exponent-mantissa split preserves the dynamic range required by activations with heavy-tailed distributions, where integer formats such as INT8 typically incur substantial accuracy loss even elaborate calibration or outlier handling. Modern accelerators amplify this benefit at the hardware level, by reusing the same Tensor Core datapath at reduced precision, FP8 matrix multiplications achieve roughly 2x throughput over BF16 on current-generation GPUs, while simultaneously halving the bandwidth pressure of weight loading cost that often dominates inference latency at small batch sizes typical of robotic control loops. Concretely, we apply *post-training quantization* (PTQ) in FP8 with per-block scaling, weights and activations are partitioned into fixed-size blocks along the reduction dimension, each assigned an independent scaling factor that absorbs local magnitude variation while keeping the per-tensor metadata overhead negligible. This granularity strikes a favorable trade-off between quantization fidelity and runtime efficiency—coarser tensor-wise scaling fails to track the channel-wise outliers characteristic of transformer activations, while finer per-element schemes introduce dequantization overhead that erodes the throughput advantage FP8 is intended to deliver.

A naive implementation of per-block FP8 quantization can erase most of these throughput gains, since the cost of standalone computation of per-block scales and casting tensors at runtime is non-trivial relative to the GEMM itself. We therefore implement the quantization operators carefully along two fronts,weights are quantized *offline* into pre-packed FP8 tensors with their scaling factors baked in, so the runtime path bears no weight-side quantization cost, and on-the-fly *activation* quantization is fused into the epilogue of the preceding operator, computing the per-block scale and casting to FP8 within the same kernel that produced the activations—avoiding a separate read/write pass over the tensor and reducing the quantization overhead to a negligible fraction of overall GEMM time.

Combined with general-purpose deployment optimizations such as CUDA Graph capture to eliminate host-side launch overhead in the per-step critical path, the full stack of optimizations described above, brings end-to-end inference to **10Hz**, meeting the latency budget required for closed-loop robotic control.

7 Experiments

7.1 Embodied Video Generation Evaluation

Evaluation Protocol. To evaluate video-generation capability in embodied settings, we assess WALL-WM as an *embodied video generator*. Given an initial observation and a language instruction, the model must predict the subsequent event while preserving object identity, robot motion, contact dynamics, and cross-view geometric consistency, so that the generated sequence remains useful for downstream control. Unless otherwise specified, WALL-WM uses a video tower initialized from Wan2.2-5B, while the Wan-series baselines are evaluated without WALL-WM's event-centric embodied training. Unlike conventional video-generation benchmarks that focus primarily on perceptual fidelity, evaluating embodied world models must reflect practical value for real embodied applications. We therefore follow the WorldArena evaluation protocol (65) to test whether the model has evolved from a generic video prior into an embodied world model prior.

Benchmark construction. We construct an Embodied Video Generation benchmark from our generalized embodied data mixture. The held-out split contains 200 representative in-distribution tasks and 50 out-of-distribution tasks, covering diverse verbs, object categories, scene layouts, camera configurations, and robot embodiments. The OOD split stresses text and compositional generalization through novel object–verb pairings, paraphrased instructions, unseen scene arrangements, and task compositions that share high-level language with the training set yet require distinct visual trajectories. Overall, this protocol tests whether the model learns an event-centric mapping from language and observation to future physical states under the next-event prediction paradigm.

Main results. As shown in Table 2, WALL-WM consistently outperforms the Wan-series baselines (Wan2.1 and Wan2.2) on embodied-relevant dimensions, including Motion Quality, Semantic Consistency, and Physical

Models	Visual Quality		Motion Quality			Semantic Consistency			Physical Plausibility			
	Image Quality	Aesthetic Quality	Dynamic Degree	Flow Score	Motion Smoothness	Subject Consist.	Background Consist.	Semantic Alignment	Interaction Quality	Perspective	Instruction Following	Trajectory Acc.
Wan2.1-1.3B (72)	**0.577**	0.389	0.199	0.061	0.619	0.476	0.522	0.857	0.219	0.819	0.308	0.214
Wan2.2-5B (72)	0.527	**0.409**	0.418	0.109	0.683	0.769	0.817	0.805	0.226	0.807	0.298	0.223
WALL-WM	0.503	0.393	**0.484**	**0.148**	**0.771**	**0.795**	**0.838**	**0.886**	**0.434**	**0.821**	**0.391**	**0.234**

Table 2 Quantitative comparisons with foundational video generation models. Our method improves the embodied-relevant metrics of Motion Quality, Semantic Consistency, and Physical Plausibility while preserving competitive visual quality. The best performing metrics are highlighted in bold.

Figure 14 Qualitative results. Our method exhibits better alignment with real-world physical laws, showing more consistent motion and physically plausible interactions, while Wan2.1 and Wan2.2 frequently generate hallucinated objects or unrealistic dynamics (e.g., contact failures, object deformation).

Plausibility. These gains suggest that large-scale embodied training, together with our training recipe, turns the inherited video prior into a stronger physical prior that preserves coherent motion and contact dynamics during generation. Beyond competitive perceptual quality, WALL-WM leads on physically grounded and interaction-oriented criteria, indicating tighter alignment with realistic manipulation dynamics. Qualitatively, generic video models frequently exhibit semantic drift, whereas our embodied world model produces more coherent and goal-consistent rollouts.

Probed Feature	Point Err(↓)	Depth Err(↓)	AUC@5 (↑)	AUC@30 (↑)
DINOv2 (58)	0.559	0.209	0.051	0.508
V-JEPA (4)	0.439	0.214	0.076	0.619
CogVideoX (81)	0.485	0.231	0.051	0.569
Aether (96)	0.501	0.249	0.054	0.571
Open-Sora2.0 (91)	0.391	0.196	0.096	0.643
WAN2.1-14B (72)	<u>0.284</u>	0.151	<u>0.200</u>	**0.736**
WALL-WM	**0.271**	**0.132**	**0.210**	<u>0.727</u>

Table 3 3D awareness benchmark on CO3Dv2 (63). We evaluate different visual representations using point error, depth error, and AUC metrics. WALL-WM achieves competitive 3D awareness, complementing its strong multi-view consistency observed in the cross-view evaluation.

Analysis. Compared with Wan2.1 and Wan2.2, WALL-WM performs consistently better on our generalized embodied benchmark. As shown in Fig. 14, improvements in Physical Plausibility, Interaction Quality, and Trajectory Accuracy indicate that large-scale embodied training turns the inherited Wan video prior into a physical prior over robot–object interaction and contact evolution. This effect is enabled by our diverse embodied data mixture, which spans varied tasks, robot configurations, camera views, and language instructions and thus exposes the model to a broader range of embodied dynamics.

WALL-WM also strengthens multi-view collaboration through cross-view attention modules, which perform cross-view information exchange inside the pretrained WAN backbone. Compared with direct multi-view concatenation, this prior-preserving design retains Wan's strengths in texture synthesis and language alignment while injecting synchronized multi-view interaction dynamics. As a result, generated rollouts are more geometrically consistent across views.

To quantitatively examine this property, we follow the VidFM-3D(35) evaluation protocol and measure multi-view consistency using cross-view correspondence error, together with 3D awareness metrics based on point and depth probing. As shown in Fig. 15 and Table 3, WALL-WM achieves strong performance on both multi-view correspondence and 3D-aware probing. In particular, the lower correspondence error indicates better cross-view geometric consistency, while the competitive 3D probe results suggest that the learned representations preserve spatial structure beyond appearance-level generation. These results demonstrate that WALL-WM not only improves visual fidelity and action generation, but also learns view-consistent and 3D-aware dynamics.

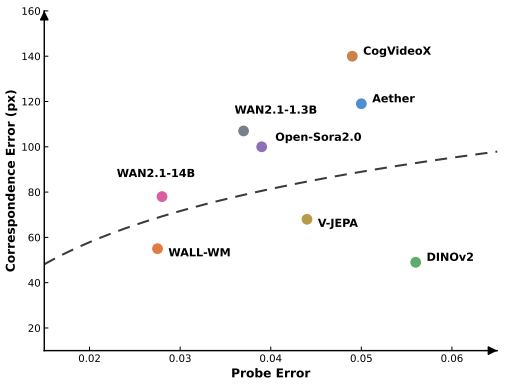


Figure 15 Multi-view consistency and 3D awareness. 3D Probe Error vs. Cross-view Correspondence Error (lower is better). WALL-WM achieves jointly low probe and correspondence errors, evidencing 3D-aware and view-consistent representations.

Our model also exhibits stronger text generalization. Wan models often capture broad nouns and verbs, but become less reliable when prompts describe new compositions of familiar manipulation concepts. WALL-WM is trained with multi-level event captions and prompt augmentation, tying language to local physical phases such as reach, contact, grasp, transfer, and release. This allows generated sequences to better follow paraphrased or compositional instructions, rather than simply matching familiar canonical descriptions.

7.2 Real-Robot Evaluation

Experimental setup. The real-robot evaluation is conducted on the high-performance tabletop bimanual robot arms from the internally developed deployment platform suite (Fig. 7). The platform uses synchronized multi-view observations and language instructions, with evaluation scenes aligned to the tabletop deployment geometry described in Sec. 4. The benchmark is organized into four complementary suites covering direct manipulation, instruction-level reasoning, fine contact control, and generalization in cluttered scenes. All models are compared using *Task Progress*, a dense 0–100 score that credits partial completion of the specified task according to task-specific rubrics listed in Table 6. We use Task Progress as the primary metric rather than binary task success because a pure success-rate criterion is often too coarse for real-robot evaluation. Because task completion is causal and thresholded, binary success can obscure meaningful differences in grounding, contact handling, and intermediate physical reasoning, and thus deviate from the actual competence reflected by the pretrained model. Task Progress provides a more continuous measure of how far the robot advances toward the specified goal, allowing near-completions, partial state changes, and meaningful intermediate progress to be reflected in the score; the rubric scores are normalized to the 0–100 scale used in the main results. We therefore report Task Progress uniformly across all real-robot suites to better characterize the practical performance of the pretrained models. The detailed task-level Task Progress scores for these experiments are recorded in Table 5. The baselines include $\pi_{0.5}$ (36), DreamZero (83), and LingBot-VA (44) when corresponding scores are available. To ensure a fair comparison, all deployed policies are evaluated under the same physical task definitions, language instructions, multi-view observation streams, scene randomization protocol, and Task Progress rubrics. Baseline policies are adapted only through their standard action interfaces for the target embodiment, and no method receives privileged state information or task-specific scoring feedback during rollout. When a baseline could not be deployed on a particular suite under this protocol, we mark the entry as unavailable rather than mixing results from a different evaluation setting. The main WALL-WM policy is evaluated in *event mode*, where the policy starts from the pretrained event-centric world-action backbone and uses the proposed text reasoning module to provide event-conditioned control. For ablation, we additionally report WALL-WM-U-Scratch, a from-scratch unified baseline trained directly in a fixed-length

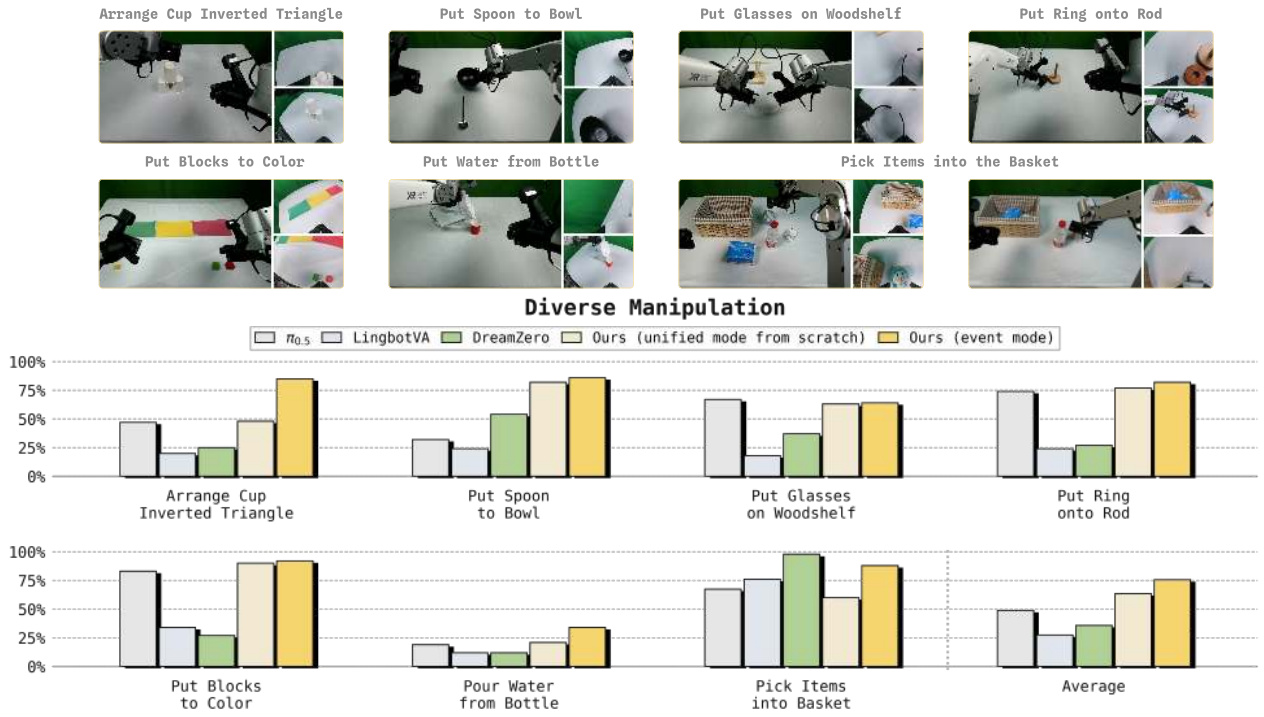


Figure 16 Real-robot Diverse Manipulation benchmark. Task Progress scores on direct tabletop manipulation tasks, including grasping, placing, pouring, spatial relocation, and color-conditioned placement.

instruction-to-action setting on the same real-robot task supervision. This baseline should not be confused with the unified inference mode introduced in Sec. 5.5.2, which operates on the event-pretrained backbone; instead, WALL-WM-U-Scratch removes the event-centric pretraining and event-conditioned reasoning pathway to test how much transferable real-robot control prior they provide.

7.2.1 Diverse Manipulation

Diverse Manipulation is designed to assess a broad range of direct physical manipulation skills under relatively explicit task goals. The task set consists of `Arrange Cup Inverted Triangle`, `Put Spoon to Bowl`, `Put Glasses on Woodshelf`, and `Put Ring onto Rod`. It further includes `Put Blocks to Color`, `Pour Water from Bottle`, and `Pick Items into Basket`. Together, these tasks test spatial arrangement, grasp–place coordination, object relocation, tool-object interaction, color-conditioned placement, pouring, and visually guided transfer. They are not primarily designed to test abstract reasoning; instead, they stress whether the policy can localize small tabletop objects, maintain stable contact transitions, and complete the intended physical state change from multi-view observations.

Results. Fig. 16 reports that event-mode WALL-WM achieves the strongest average Task Progress on the Diverse Manipulation suite, reaching 75.86 compared with 63.00 for WALL-WM-U-Scratch, 55.64 for $\pi_{0.5}$, 39.97 for DreamZero, and 29.71 for LingBot-VA. The gains are most visible on tasks that require complete object-state transitions rather than a single short reach, such as arranging cups into an inverted triangle, placing a spoon into a bowl, putting a ring onto a rod, moving blocks to target colors, and pouring water from a bottle. WALL-WM-U-Scratch is also competitive on several direct manipulation tasks, which suggests that the architecture can learn common motor patterns from task supervision alone. However, event mode improves the average score by 12.86 points over WALL-WM-U-Scratch and remains more consistent across the suite, suggesting that event-centric pretraining may provide a reusable physical prior for grasp, transfer, alignment, and release phases. The comparison also reveals meaningful task-level variation: $\pi_{0.5}$ slightly leads on shelf placement, and DreamZero obtains the highest score on basket picking. Thus, the main advantage of event-mode WALL-WM lies not in winning every individual task, but in delivering the strongest overall Task Progress across a diverse set of tabletop manipulation skills.

Figure 17 Real-robot Reasoning Manipulation benchmark. Task Progress scores on tasks requiring semantic classification, ordering, matching, and instruction-conditioned selection.

7.2.2 Reasoning Manipulation

Reasoning Manipulation examines instruction grounding beyond single object motion. The task set consists of `Sort Headphone`, `Classify Items as Shape`, and `Press Button in Order`. It further includes `Pair Up Items` and `Pick Fruits into Basket`. These tasks require category grounding, relational matching, ordered execution, and instruction-conditioned object selection. Unlike Diverse Manipulation, the same scene can admit multiple plausible actions, so the policy must infer which object, category, relation, or sequence is intended by the language instruction before executing the corresponding manipulation.

Results. Fig. 17 indicates that event-mode WALL-WM obtains the strongest average result on reasoning-heavy tasks, reaching 71.60 compared with 59.50 for WALL-WM-U-Scratch, 56.40 for $\pi_{0.5}$, 32.70 for DreamZero, and 31.60 for LingBot-VA. The stable gain over WALL-WM-U-Scratch suggests that the event-mode improvement is not only due to the architecture, but is also associated with the pretrained manipulation backbone and the introduced language-guided reasoning mechanism. In our real-robot experiments, this mechanism is implemented with a fine-tuned Qwen3.5-VL-9B model that maps broad task requirements and current multi-view observations into concrete next-event descriptions, consistent with the event-mode inference recipe in Sec. 5.5.1. These next-event descriptions provide explicit event-centric control targets for the pretrained world-action backbone, which may help when the instruction changes the required event structure, such as ordered button pressing and fruit selection. WALL-WM-U-Scratch remains competitive on recognition-oriented cases such as sorting and classification, and $\pi_{0.5}$ performs best on relational pairing. Thus, the main advantage of event-mode WALL-WM is its stronger overall balance across category grounding, ordered execution, and instruction-conditioned selection, supported by language-guided conversion from high-level instructions to executable event descriptions.

7.2.3 Dexterous Manipulation

Dexterous Manipulation emphasizes contact-sensitive behaviors that are difficult to capture with coarse semantic instructions alone. The figure visualizes two fine-grained tasks: `Put Stationery in Case`, and `Insert Wireline`. The averaged Task Progress score is reported on the scored insertion and constrained-placement tasks. These tasks require narrow-tolerance insertion, precise end-effector alignment, careful contact timing, and, in the fan-unboxing example, longer object-handling sequences with packaging constraints. Small pose errors can prevent progress even when the high-level object selection is correct, making the suite a stress test for local action precision rather than semantic understanding alone.

Results. Fig. 18 shows that event-mode WALL-WM obtains the highest averaged Task Progress on the Dexterous

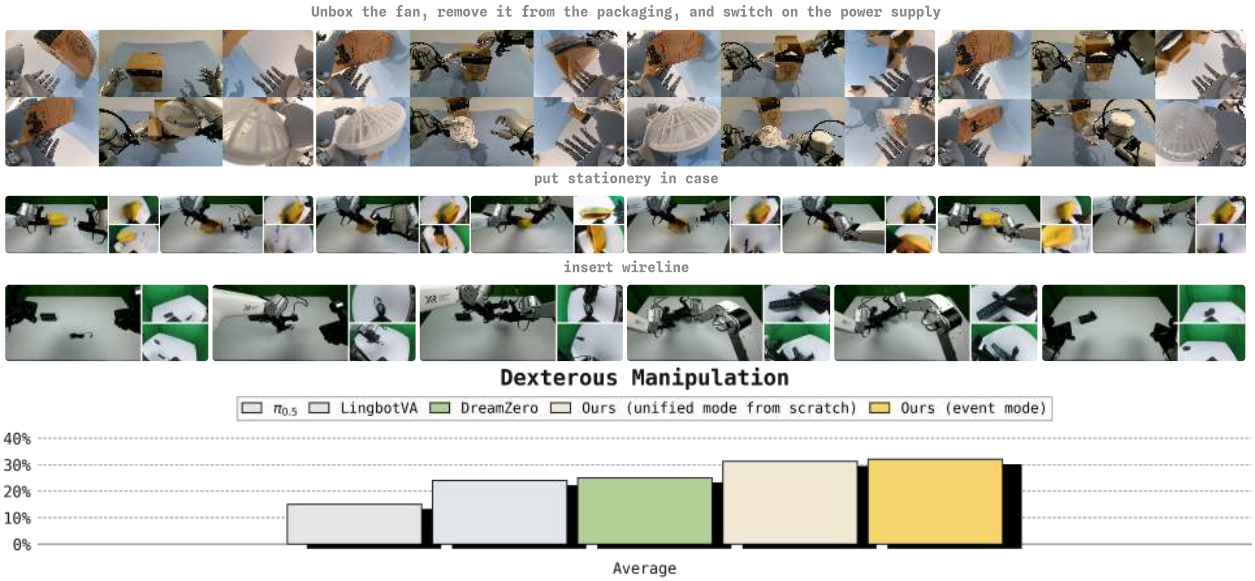


Figure 18 Real-robot Dexterous Manipulation benchmark. Average Task Progress on contact-sensitive tasks that require precise insertion and fine object handling.

Manipulation suite, reaching 32.00, closely followed by WALL-WM-U-Scratch at 31.25. Both modes improve over the averaged scores of $\pi_{0.5}$, LingBot-VA, and DreamZero, suggesting that the pretrained video-action backbone may provide a useful local dynamics prior for contact-rich motion. However, the gap between event mode and WALL-WM-U-Scratch is small, indicating that these tasks are less limited by high-level event decomposition and more constrained by low-level pose accuracy, contact timing, and narrow-tolerance alignment. At the task level, both WALL-WM modes achieve the best score on `Insert Wireline`, while DreamZero performs better on `Put Stationery in Case`. Overall, the relatively low absolute scores show that dexterous real-robot manipulation remains highly challenging, especially for precise insertion and constrained placement where small contact errors can prevent task progress.

7.2.4 Generalization

Generalization is designed to assess instruction-conditioned transfer in a complex tabletop scene rather than in isolated task-specific resets. During evaluation, multiple objects remain in the scene and different instructions are issued in randomized order, requiring the policy to ground the current command against the shared scene state. The tasks include `Place Plates into Storage Slots`, `Cover Pot with Lid`, `Push Cleaning Cloth to Table Edge`, and `Insert Screwdriver into Cup`. The setting targets scene-level grounding, instruction switching, robustness to distractors, and compositional transfer. The robot must avoid overfitting to a fixed task order and instead select the action implied by the current instruction under visual clutter and shared object context.

Results. Fig. 19 reports that event-mode WALL-WM obtains the highest average Task Progress on the Generalization suite, reaching 53.75 compared with 28.50 for DreamZero, 24.00 for $\pi_{0.5}$, and 18.50 for WALL-WM-U-Scratch. The improvement is most visible on `Place Plates into Storage Slots`, `Push Cleaning Cloth to Table Edge`, and `Insert Screwdriver into Cup`, where the policy must ground the current instruction among multiple objects and select the corresponding manipulation event. This trend is consistent with event-mode inference: the language-guided reasoning module converts the current command into a next-event description, while the pretrained world-action backbone executes reusable transitions such as approach, align, insert, push, cover, and release. At the same time, the results are not uniform across all tasks; WALL-WM-U-Scratch performs better on `Cover Pot with Lid`, suggesting that some visually direct motions can still be learned from task-level supervision without explicit event decomposition. Overall, the suite indicates that event-conditioned execution can improve robustness under instruction switching and shared scene context, while generalization in cluttered real-robot settings remains challenging.

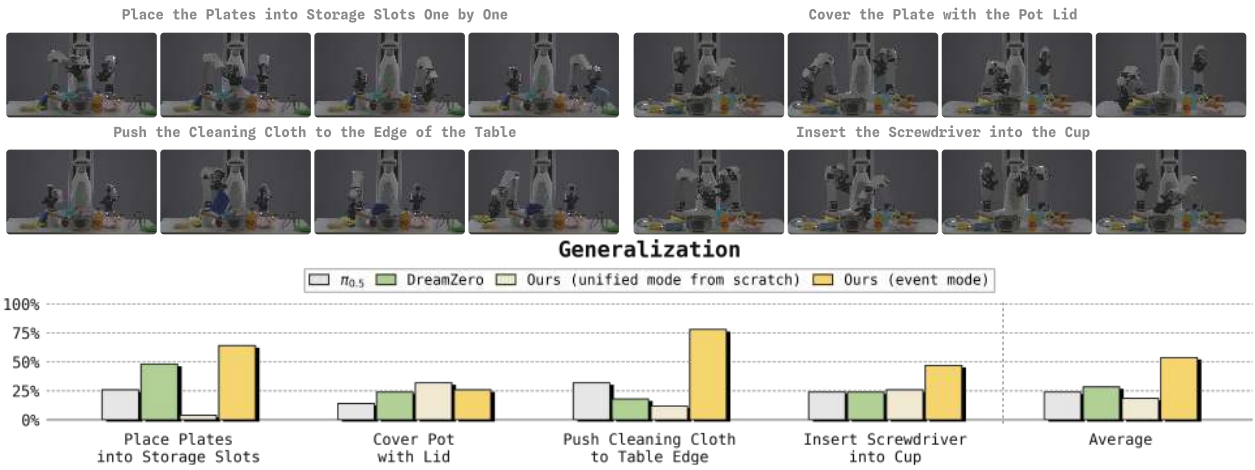


Figure 19 Real-robot Generalization benchmark. Task Progress scores in a complex tabletop scene where different instructions are issued in randomized order, testing scene grounding and compositional transfer.

Reasoning Manipulation Task	Base	Event	Generalization Task	Base	Event
`Sort Headphone`	55	84	`Place Plates into Storage Slots`	30	64
`Classify Items as Shape`	40	78	`Cover Pot with Lid`	18	26
`Press Button in Order`	0	64	`Push Cleaning Cloth to Table Edge`	26	78
`Pair Up Items`	8	36	`Insert Screwdriver into Cup`	14	47
`Pick Fruits into Basket`	60	96	–	–	–
Average	**32.6**	**71.6**	**Average**	**22.0**	**53.75**

Table 4 Ablation on event-conditioned execution and VI-SA. Base denotes a pretrained fixed-length unified baseline without View-Interaction Self-Attention, while Event denotes pretrained event-mode WALL-WM. All entries report Task Progress.

7.2.5 Ablation on Event and View Modeling

To examine the contribution of event-conditioned execution and cross-view modeling, an additional pretrained baseline is evaluated on the reasoning and generalization suites. Different from the WALL-WM-U-Scratch baseline reported above, this baseline starts from the pretrained backbone but removes View-Interaction Self-Attention (VI-SA) and uses instruction-conditioned fixed-length unified decoding. Thus, the comparison controls for the presence of pretraining while removing the two design choices that most directly support multi-view scene grounding and variable-length event execution. Because the baseline differs from event mode in both VI-SA and the execution format, this ablation should be interpreted as measuring their combined effect rather than assigning the full gain to either component alone.

Results. Table 4 indicates that pretrained event-mode WALL-WM improves over the pretrained unified baseline without VI-SA on both evaluated suites. On Reasoning Manipulation, the average Task Progress increases from 32.6 to 71.6. The baseline retains some recognition capability from pretraining, as reflected by nontrivial scores on sorting and fruit selection, but it is less reliable when the instruction requires explicit temporal ordering or relational execution. For example, `Press Button in Order` improves from 0 to 64, while `Pair Up Items` improves from 8 to 36, although the latter remains challenging in absolute terms. On Generalization, the average score increases from 22.0 to 53.75. The gains are most visible on tasks such as `Push Cleaning Cloth to Table Edge` and `Insert Screwdriver into Cup`, where the robot must identify the instruction-relevant object and execute the corresponding semantic transition in a shared scene. Overall, these results suggest that pretrained representations alone are not sufficient for these settings, and are consistent with the combined benefit of cross-view interaction and event-conditioned execution for robust reasoning and generalization.

Summary. Across the four real-robot suites, event-mode WALL-WM achieves the strongest overall Task Progress.

The Diverse Manipulation results suggest that event-centric pretraining can supply a useful physical control prior for common tabletop skills, while the Reasoning Manipulation results indicate that coupling this prior with a text reasoning module may improve instruction grounding for category, order, and relation-based tasks. The Dexterous Manipulation suite highlights the remaining difficulty of fine contact control, while also suggesting that video-action pretraining can benefit local contact-sensitive motion. Finally, the Generalization suite indicates that WALL-WM can reuse pretrained event knowledge under cluttered scenes and randomized instructions, where WALL-WM-U-Scratch is less reliable. The event-and-view ablation is consistent with the view that pretraining is most effective when paired with cross-view interaction and variable-length event execution, rather than used only as a fixed-length instruction-conditioned policy. The consistent gap between event mode and WALL-WM-U-Scratch supports the hypothesis that pretrained event-centric world-action modeling provides transferable physical and semantic priors that are difficult to recover from task-level real-robot supervision alone.

8 Discussion

Scaling, latency, and the objective of generalization. Our current WALL-WM family spans model sizes from below 10B parameters to the tens-of-billions regime. Across this range, we observe a consistent trend: larger models improve both action precision and out-of-distribution generalization, especially on tasks that require fine contact timing, long-horizon state tracking, and compositional instruction grounding. This trend raises a broader question for embodied foundation models. The field has often treated real-time latency as a first-order constraint, because robotic deployment ultimately requires closed-loop execution. However, if latency is enforced too early, it may cap the reachable performance ceiling before the model has learned a sufficiently general world-action prior. We therefore view scale-driven generalization as the primary frontier for general-purpose WAMs, while treating latency reduction as a comparatively more determinate engineering problem. Distillation, quantization, speculative or streaming execution, and systems-level overlap can progressively reduce latency once a strong model exists; recovering a lost generalization ceiling from an under-scaled model is much less straightforward.

Beyond success rate as the sole evaluation target. Real-robot success rate remains the most direct measure of embodied competence, but it is also a coarse and noisy metric. A single failure can be caused by perception, language grounding, contact dynamics, hardware variation, reset quality, or stochastic environment conditions, and these causes are not uniformly distributed across tasks or evaluation runs. As a result, success rate alone has limited resolution for large-scale pretraining: it is expensive, slow, and often weak at identifying which part of the model improved. We are therefore exploring more efficient evaluation protocols for pretraining-scale iteration, including finer-grained per-event scoring, dense progress measures, human correction and relabeling during evaluation, and general-purpose evaluation models that predict robot performance from generated video, action traces, and intermediate states. Such an evaluator does not need to perfectly replace real hardware. It only needs to maintain a reliable positive correlation with downstream robot performance, so that model and data decisions can be screened cheaply before the most promising candidates are validated on real robots.

Event-centric instruction generalization as an interface to agents. We view per-event textual generalization as a bridge between embodied control and broader agentic intelligence. Current embodied foundation models often spend trainable parameters on understanding and reasoning over high-level instructions, but the rapid progress of frontier multimodal models and agents (57, 17, 1) forces a boundary question: should a high-frequency manipulation model compete for parameters on over-high-level task reasoning that a general agent can already perform? In many embodied benchmark designs, this boundary remains underspecified. Our preference is to factor information from high-level tasks into per-event textual descriptions, and then from per-event language into vision and action. Under this decomposition, full-domain action-text instruction generalization over semantic events becomes the highest-priority objective for the embodied model, while excessive concern over the reasoning cost or latency of the upstream agent may itself become a bottleneck to the emergence of embodied intelligence.

Contributors

WALL-WM is a collaborative effort of the X Square Robot team. The full contributor list is given below; * denotes core contributors, † denotes the project lead, and ‡ denotes the corresponding author.

Shalfun Li*†, Victor Yao*, Charles Yang*, Truth Qu*, Regis Cheng*, Ryan Yu*, Howard Lu*, Newton Von*, Vincent Chen*, Yohann Tang, Maeve Zhang, Ellie Ma, Gody Li, Sage Yang, Lorien Shu, J.W. Gao, Ethan Chen, Colin Ye, Rain Sun, Elise Mon, PS Zhang, Neo Li, Lily Li, James Wang, Ping Yang, Chris Pan, Lucy Liang, Hang Su, Roy Gan, Hao Wang‡, Qian Wang.

9 Appendix

9.1 Modeling Choices: Pixel-Space, Native T2V, and a Capacity-Balanced Dual-Tower

Three design choices, taken together, place WALL-WM in the design space. The recent VLA literature tends to collapse them into a single "pixel vs. latent" question, but they are better read as three distinct trade-offs with separate logic, and WALL-WM takes the more demanding side on each axis. We discuss the three in turn.

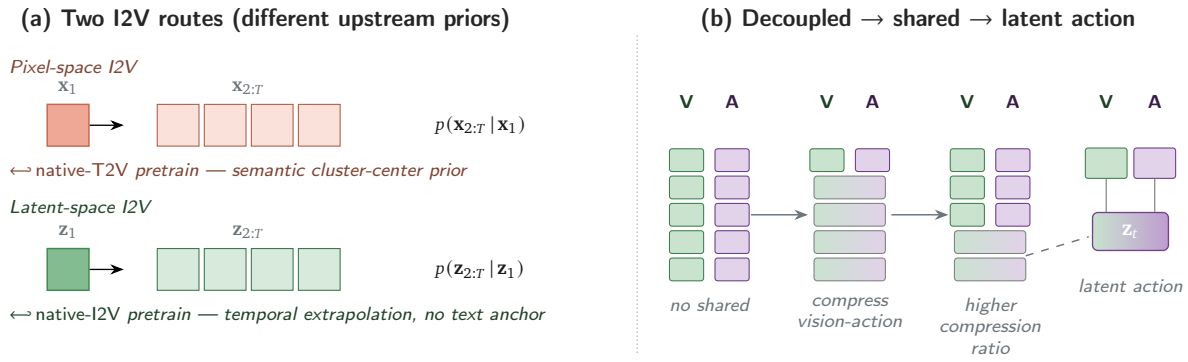


Figure 20 Two modeling-choice axes that place WALL-WM in the design space. *(a) Pretraining-route axis.* Pixel-space (top, orange) vs. latent-space (bottom, green) backbones both perform the same downstream I2V job, but inherit qualitatively different upstream priors—a *native-T2V* cluster-center prior on the pixel side, a denser *native-I2V* temporal-extrapolation prior on the latent side. WALL-WM keeps the pixel + native-T2V combination (Section 9.1). *(b) Dual-tower spectrum.* This is an abstract illustration rather than a concrete architectural sweep: the latent video-action representation is implicitly distributed across the DiT parameters, and under certain capacity-sharing regimes, intermediate features of the coupled video-action DiT become operationally equivalent to the narrow notion of a latent action.

(i) Pretrained pixel-space prior vs. pretrained latent-space prior. The first axis is largely empirical: pixel-space video foundation models have, by now, been pretrained on far broader and far thicker corpora than any latent-space counterpart available off the shelf, and the resulting weights—redundant as they often are—encode a strong prior over the visual world. Setting aside this historical asymmetry, the latent side still carries an intrinsic advantage worth naming: by inducing a smoother and more structured representation manifold, latent training reduces reliance on stochastic score-based generation and leaves a flatter, more tractable optimization landscape behind. For a video + action world model whose downstream burden is dynamics and control, however, the sheer breadth of the pixel-space pretraining prior dominates: initialising from a pixel-space backbone is therefore a pragmatic choice rather than a stance against latent representations.

(ii) Native T2V vs. native I2V, even when both are trained from scratch. The second axis is easy to miss but consequential once noticed: most of today's pixel-space video foundation models—including those formally trained under an I2V objective—are themselves built on top of an upstream *native-T2V* pretraining map, while a separate line of work (e.g. V-JEPA-2 (2), LeWorldModel (54)) starts directly from *native-I2V* mappings. Neither route admits a clean winner; the relevant trade-off is the following. A native-I2V objective tracks the temporal evolution of the visual signal more tightly than text-to-vision ever can—given the previous frame, the visual continuation is densely supervised at every pixel—which is a genuine strength. A native-T2V objective, in turn, brings in a prior of a different flavour. Text behaves like a low-dimensional *cluster-center label* over the high-dimensional manifold of plausible visual futures: a T2V objective therefore implicitly asks the network to first commit to the semantic class of the trajectory it is about to render and only then paint it, which is a form of semantically anchored visual self-supervision delivered for free at internet scale. I2V removes that anchor. With nothing to commit to up front, the easiest minimum of the I2V loss is high-bandwidth pixel-extrapolation conditioned on the previous frame—closer in spirit to a learned optical flow than to a world model—and any gradient that would teach high-level physical regularities flows through that route only very thinly. Higher-level structure (intent, object identity, sub-task boundaries) is correspondingly only weakly captured even when scaling is generous. WALL-WM remains on the T2V side, so that this cluster-center prior is paid for once and inherited by everything downstream (Figure 20a).

(iii) Latent-action vs. dual-tower video + action. The two paradigms appear opposed at first sight: one compresses dynamics into an explicit, often discrete, action code, while the other places two DiT-scale towers side by side. On closer inspection, however, they converge. Latent-action methods explicitly bottleneck the next observation into a vision-aligned, implicit dynamic-action code that the downstream policy must decode; a dual-tower video + action design, once the ratio between cross-tower shared capacity (the layer-wise coupling CA, AdaLN modulation, the shared frame-index PE) and tower-private capacity (the per-tower FFN width, the action-only state cross-attention) settles at the right operating point, performs essentially the same compression. The shared subspace *is* a latent action, except that it is emergent and end-to-end learned rather than imposed through an architectural bottleneck whose width and codebook have to be guessed in advance. The dual-tower formulation is therefore better read as a more permissive form of latent action than as its competitor: tightening the shared sub-block recovers any explicit-bottleneck regime, while loosening it leaves room for extra width when the true dynamics outgrow any latent dimensionality picked by hand. Formally, let $\mathbf{h}_t^{\text{shared}} \in \mathbb{R}^d$ denote the cross-tower bottleneck activation through which V information reaches the A tower, and let $\mathbf{z}_t$ be the explicit latent-action code of an LAPA-style encoder/codebook/decoder; when d matches the codebook width, the two carry the same information bottleneck and are interchangeable up to the discreteness of $\mathbf{z}_t$ (Figure 20b).

Together these three choices place WALL-WM on a pixel-space initialization, a native-T2V objective, and a dual-tower with deliberately sized shared and private capacity.

9.2 Temporal-Robust Reconstruction in the Wan VAE

A practical question is whether the Wan VAE's temporal downsampling relies on strong short-range temporal correlations. We stress-test this assumption with deliberately adversarial inputs: frames sampled far apart in time, and even heterogeneous still images stacked along the temporal axis as if they formed a video. The VAE still reconstructs these sequences successfully, suggesting that its compression is not heavily biased toward exploiting local temporal redundancy. This property benefits WALL-WM: even frame-skipped embodied videos need not resemble smooth web-video snippets for the VAE to preserve per-frame visual content.

9.3 Visual-Semantic and Spatial Strength of the Inherited Wan Features

This subsection explains why the inherited Wan features sit at the bottom of the conditioning stack as an *immovable substrate*, rather than as just another swappable encoder. Although Wan is trained with a video-generation objective (instead of a discriminative one), its hidden-state geometry encodes both (i) object/instance-level visual-semantic structure and (ii) 3D, camera-consistent spatial structure. On the spatial axis, it is in practice stronger than canonical contrastive image-pretraining features.

Cross-view PCA: a direct visual diagnostic. We probe this on a synchronized *ego / left-wrist / right-wrist* triplet from the deployment camera rig. We extract intermediate Wan DiT features at the same denoising step from all three views, project them with a *shared* PCA basis fit jointly across the triplet (Figure 21, middle row), and overlay the resulting PCA coloring onto the original frames (bottom row). Three patterns emerge. First, semantically equivalent surfaces—the manipulated object, the operator's hand, and the empty bowl—collapse to a small set of PCA components and receive *consistent* colors across views, despite the three cameras sharing little raw-pixel overlap. Second, the coloring tracks underlying physical surfaces rather than view-specific image gradients: the same object largely preserves its PCA signature under the camera-pose change between left- and right-wrist. Third, per-token cross-view cosine consistency stays at ~ 0.97 for every pair of views (see the meta-data strip in Figure 21). Under the same protocol, ViT-SigLIP and self-supervised image features extracted from the same triplet yield reproducibly lower consistency; the gap concentrates on geometry-sensitive tokens (object edges, contact regions) rather than background.

9.4 Can KV-Cache Streaming Solve the Limits of V-A Temporal Alignment?

A recurring difficulty in vision–action world models is *temporal alignment*. Language typically specifies a semantic intent whose temporal support is underspecified, while video and action unfold over many frames (Figure 2, Section 3.1). Recent work—including DreamZero (83) and LingBot-VA (44)—adopts KV-cache–compatible rollouts to stream visual/action context across chunks. This helps retain historical state and can partially reduce V–A alignment drift during long-horizon prediction.

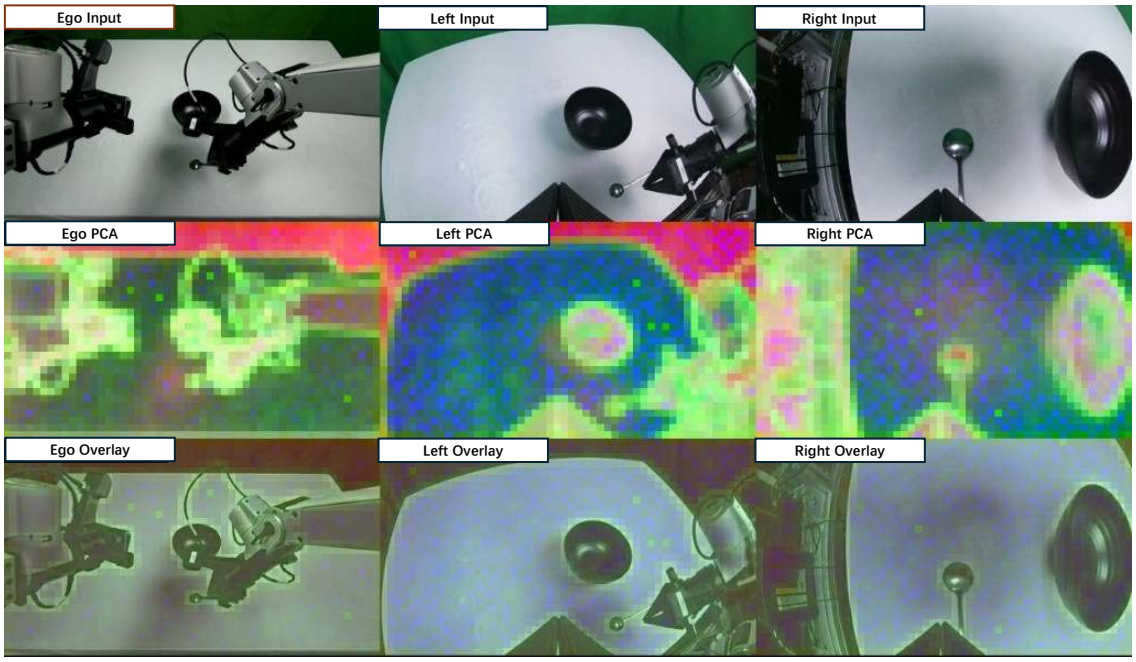


Figure 21 Tri-view PCA consistency of Wan generative features. For a synchronized *ego / left-wrist / right-wrist* triplet (top: raw inputs), we extract intermediate Wan2.1 DiT features and project them with a PCA basis fit jointly across the three views (middle); the bottom row overlays the resulting PCA coloring onto the input frames. Semantically equivalent surfaces (the manipulated object, the operator's hand, the empty bowl) receive consistent PCA signatures across cameras despite minimal raw-pixel overlap, and per-token cross-view cosine consistency stays at ~ 0.97. Read alongside (78, 34), this provides a concrete visual rationale for treating the inherited Wan features as a visual-semantic *and* spatial substrate, rather than as a generic image prior.

However, the fix remains *partial*. These systems still train and infer with *fixed-horizon* future prediction: the model generates a predetermined number of frames after the current anchor, rather than stopping at the instruction's semantic endpoint.

9.5 Qualitative Overview: Pre-training Data and Deployed Execution

This section provides qualitative visual examples for the pre-training data, real-robot evaluation scenes, and deployment-time prediction. Figure 22 shows representative pre-training tasks and segment-level captions; Figure 23 shows the scene diversity used in real-robot evaluation; and Figure 24 compares generated future video with the corresponding real execution.

Task	$\pi_{0.5}$	LingBot-VA	DreamZero	WALL-WM-U-Scratch	WALL-WM-E
Diverse Manipulation					
`Arrange Cup Inverted Triangle`	47	20	25	48	85
`Put Spoon to Bowl`	32	24	54	82	86
`Put Glasses on Woodshelf`	67	18	37	63	64
`Put Ring onto Rod`	74	24	27	77	82
`Put Blocks to Color`	83	34	27	90	92
`Pour Water from Bottle`	19	12	12	21	34
`Pick Items into Basket`	67.5	76	97.8	60	88
Average	**55.64**	**29.71**	**39.97**	**63.00**	**75.86**
Reasoning Manipulation					
`Sort Headphone`	41	56	66	84	84
`Classify Items as Shape`	55	30	36	82	78
`Press Button in Order`	31	8	3	18	64
`Pair Up Items`	77	4	13.5	43.5	36
`Pick Fruits into Basket`	78	60	45	70	96
Average	**56.40**	**31.60**	**32.70**	**59.50**	**71.60**
Dexterous Manipulation					
`Insert Wireline`	18	28	24	42	42
`Put Stationery in Case`	12	20	26	20.5	22
Average	**15.00**	**24.00**	**25.00**	**31.25**	**32.00**
Generalization					
`Place Plates into Storage Slots`	26	–	48	4	64
`Cover Pot with Lid`	14	–	24	32	26
`Push Cleaning Cloth to Table Edge`	32	–	18	12	78
`Insert Screwdriver into Cup`	24	–	24	26	47
Average	**24.00**	**–**	**28.50**	**18.50**	**53.75**

Table 5 Detailed real-robot Task Progress scores. Task-level Task Progress is reported for all real-robot evaluation suites. WALL-WM-U-Scratch and WALL-WM-E denote unified mode from scratch and event mode, respectively. A dash indicates that the corresponding baseline was not evaluated.

#	Task	Image	Instruction	Image	Instruction	Image	Instruction	Image	Instruction
Easy									
1	Pick Up Cup		The right arm puts the orange cup on the light green plate		The right arm moves to the side of the plate and pushes the green...		Move the left arm next to the blue cup and grasp the blue cup		The right arm puts the orange cup on the light green plate
2	Push Cup		The right arm moves next to the table, pushes the black cup, then...		The right arm places the cup on the silver saucer		The right arm moves to the right		The right arm moves next to the brown cup and picks it up.
3	Pick Apple		Move the right arm to the side of the table to grasp the pink apple on the...		The right arm puts the apple into the drain basket		The right arm moves to the right side of the table		The right arm moves to the table and picks up the apple
4	Insert Flowers (Embod-Free)		Insert the light green flower bouquet into the slender vase, and...		Insert the pink rose, red rose and green matching flowers into the...		Insert the green flowers, pink roses and red roses into the...		Insert the light green flower bouquet into the tall vase, and...
5	Move Items (Embod-Free)		Push the blue rag and various skin care bottles on the table to the...		Move the scattered skin care products, facial cleanser and blue...		Move the blue rag, various skin care bottles and tubes on the table to...		Move all small items in the middle of the table to the left...
Hard									
6	Arrange Cups & Plates (5 segs)		The left arm places the white round plate on the table to the right...		The right arm places the pink plate on the tabletop to the...		The left arm places the pink ceramic mug on the white round plate		The right arm places the orange mug on the beige square plate
7	Assemble Chair (6 segs)		The right arm moves to the yellow cube, picks it up and places...		The right robotic arm moves to the lower right of the table to pick up...		The right arm of the robot moves to the middle of the table, and places...		The right arm moves to the side of the white rectangular block,...
8	Open Takeout Box (5 segs)		Both arms move to the side of the takeout box		The right arm moves next to the Meituan takeout box to tear off...		The right arm opens the lid of the Meituan food delivery box.		The left arm secures the takeout box, and the right arm...
9	Adjust Glasses (11 segs)		The left arm picks up the gray rolled cloth, moves to the front of the...		The left and right arms move next to the leather sheet. The left arm holds...		The right arm moves to the side of the left arm and unfolds the...		The left arm and the right arm place the glasses onto the glasses...
10	Fold Towel (Embod-Free)		Fold the black T-shirt on the table neatly		Fold the orange T-shirt on the table neatly		Fold the blue towel neatly on the table		Fold the yellow towel on the table neatly
11	Put Items in Drawer (Embod-Free)		Put the Rubik's cube, snack bag and black strap on the table into the...		Fold the blue rag and put it into the desk drawer together with the...		Put all the items on the desktop into the open drawer		Neatly organize and put all items on the table into the open drawer
Long Horizon									
12	Sort Sachets (64 segs)		The left arm moves to the left beige storage basket and picks up the white...		The left arm moves to the wooden box to grab the yellow sachet		The right arm puts the small white object into the pink storage...		The right robotic arm adjusts the white objects inside the pink...
13	Deal Mahjong Tiles (78 segs)		Right arm moves to the table side to grasp mahjong tiles		The right arm moves to the side of the mahjong tiles on the table...		The left robotic arm moves next to the green-white mahjong tile on...		The right arm moves to the mahjong tile and flips the...
14	Fold Clothes (61 segs)		The right arm takes the orange clothes out of the basket and places...		The left arm moves to the side of the clothes on the table		The right arm moves next to the brown fabric on the table		Two arms grasp the two corners of the white T-shirt, unfold it and lay...
15	Dispose Trash (Embod-Free)		Throw all the waste packaging bags, tissues, empty plastic...		Discard all the garbage on the table into the nearby trash can		Throw all the rubbish on the table into the dustbin		Put the snack packaging bags, tissues and paper cups on the table...
16	Put Snacks (Embod-Free)		Put all the snacks on the table into the middle plate		Put all the snacks on the table into the middle plate		Put all the snacks on the table into the white plate in the middle		Put all the snacks on the desktop into the middle plate

Figure 22 Pre-training Tasks Overview. Representative pre-training tasks with four scene or event variations. Rows are grouped by difficulty, and task names in **blue** indicate embodiment-free examples. Instructions are taken from human segment-level captions.

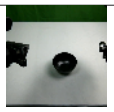
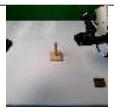
#	Task	Image	Instruction	Image	Instruction	Image	Instruction	Image	Instruction
1	**Put Spoon into Bowl**		The robot picks up the child's plastic spoon and places it into the bowl.		The robot picks up the stainless-steel spoon and places it into the bowl.		The robot picks up the stainless-steel spoon and places it into the bowl.		The robot picks up the child's plastic spoon and places it into the bowl.
2	**Put Glasses on Shelf**		The robot grasps the child eyeglass frame with both grippers and places them onto the wooden shelf.		The robot grasps the safety goggles with both grippers and places them onto the wooden shelf.		The robot grasps the adult eyeglass frame with both grippers and places them onto the wooden shelf.		The robot grasps the safety goggles with both grippers and places them onto the wooden shelf.
3	**Put Items into Drawers**		The robot pulls open the top drawer with its right gripper and places the pencil inside.		The robot pulls open the middle drawer with its right gripper and places the toy car inside.		The robot pulls open the bottom drawer with its right gripper and places the bluetooth earbuds inside.		The robot pulls open the top drawer with its right gripper and places the eraser inside.
4	**Pick Fruits into Basket**		The robot picks up the pear from the table and places it into the basket.		The robot picks up the grapes from the table and places it into the basket.		The robot picks up the banana from the table and places it into the basket.		The robot picks up the apple from the table and places it into the basket.
5	**Sort Headphones**		The robot finds the wireless bluetooth earbuds in the cluttered pile and places them into the electronics sorting bin.		The robot finds the wired in-ear earphones in the cluttered pile and places them into the electronics sorting bin.		The robot finds the over-ear headphones in the cluttered pile and places them into the electronics sorting bin.		The robot finds the wired in-ear earphones in the cluttered pile and places them into the electronics sorting bin.
6	**Put Ring onto Rod**		The robot picks up the square ring and slides it onto the vertical rod.		The robot picks up the circular ring and slides it onto the vertical rod.		The robot picks up the square ring and slides it onto the vertical rod.		The robot picks up the circular ring and slides it onto the vertical rod.
7	**Pour Water into Cup**		The robot tilts the kettle and pours water into the blue paper cup.		The robot tilts the kettle and pours water into the blue mug.		The robot tilts the kettle and pours water into the red glass cup.		The robot tilts the kettle and pours water into the red paper cup.
8	**Arrange Cups (Triangle)**		The robot places the first inverted white paper cup at the center of the table.		The robot places the second inverted red mug next to the first one.		The robot stacks the third inverted white mug on top of the two cups below.		The robot places the first inverted red paper cup at the center of the table.
9	**Put Clothes in Hamper**		The mobile-base robot drives up to the hamper and drops the shirt grasped from the sofa into it.		The mobile-base robot returns to the hamper and places the pants picked from the sofa inside.		Both arms cooperate to place a pair of socks gathered from the sofa into the hamper.		The mobile-base robot drives up to the hamper and drops the shirt grasped from the sofa into it.

Figure 23 Real-Robot Evaluation Setup. Representative real-robot evaluation scenes. Each row shows one task with four scene variations and the corresponding English sub-instructions.

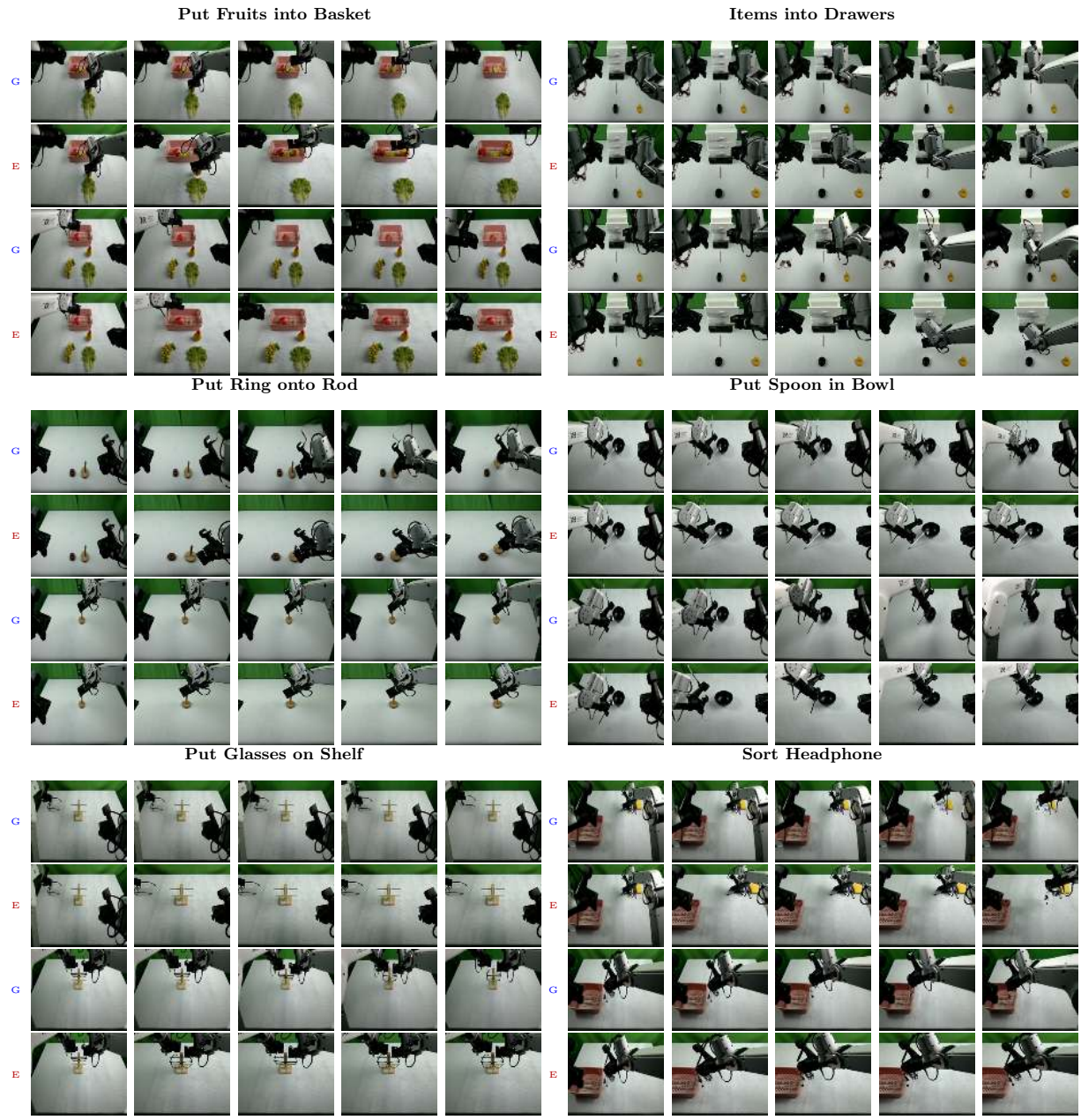


Figure 24 Generated Video vs. Real Execution. Qualitative comparison between generated future video and real-robot execution. G denotes generated frames and E denotes execution frames.

Task	Description	Scoring Rule
Diverse Manipulation		
`Arrange Cup Inverted Triangle`	Invert 3 cups in a triangle: 2 bottom, 1 top	Per cup: pick up (1) + position (1–3); top (3), retract (1)
`Put Spoon to Bowl`	Move bowl to center, place spoon in bowl	Bowl (2), move (2), spoon (2), place in bowl (3), retract (1)
`Put Glasses on Woodshelf`	Pick up glasses, adjust, place on rack	Pick up (2), center (2), adjust (2), rack (3), retract (1)
`Put Ring onto Rod`	Place a ring object onto a vertical pole	Grasp ring (3), move to pole (3), stack (3), retract (1)
`Put Blocks to Color`	Place RGB blocks onto matching color patches	3 pts per correct, 3 blocks, retract (1)
`Pour Water from Bottle`	Pour water from bottle into cup	Move cup (2), bottle (2), pour (3), set down (2), retract (1)
`Pick Items into Basket`	Pick up 4 objects and place into basket	Per object: pick up (1) + place (1.25); 4 objects, retract (1)
Reasoning Manipulation		
`Sort Headphone`	Find earphones in clutter and place them in the basket	Identify (2), pick up (3), place (4), retract (1)
`Classify Items as Shape`	Sort 3 objects by shape to target positions	3 pts per correct, 3 objects, retract (1)
`Press Button in Order`	Press 3 buttons in the instructed order	3 pts per correct press, 3 buttons, retract (1)
`Pair Up Items`	Match object pairs and place them at target positions	Per pair: grasp (2) + place (2.5); 2 pairs, retract (1)
`Pick Fruits into Basket`	Place specified fruits into basket	3 pts per correct fruit, 3 fruits, retract (1)
Dexterous Manipulation		
`Insert Wireline`	Bimanual: left picks cable, passes to right, inserts	Left grasp (3), right receive (3), insert (3), retract (1)
`Put Stationery in Case`	Bimanual: open zipper, insert items, close	Open (3.5), 1 pt per item (3), zip (1.5), retract (1)
Generalization		
`Place Plates into Storage Slots`	Pick up plates and place them into the instructed storage slots	Pick up (2), align with slot (3), place into slot (4), retract (1)
`Cover Pot with Lid`	Pick up pot lid and place on pot	Approach (2), pick up lid (3), cover (4), return (1)
`Push Cleaning Cloth to Table Edge`	Pick up cloth and wipe or push grease stains to the table edge	Approach (2), pick up (2), wipe all (5), return (1)
`Insert Screwdriver into Cup`	Pick up the screwdriver and insert it into the cup	Identify (2), pick up (3), insert into cup (4), retract (1)

Table 6 Real-robot Task Progress rubrics. Each task is scored on a 10-point Task Progress scale with partial credit assigned to observable intermediate steps. The final reported score is normalized to the 0–100 Task Progress values used in the main evaluation.

References

[1] Anthropic. Claude 3.7 sonnet. https://www.anthropic.com/news/claude-3-7-sonnet, 2025.

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[3] Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning vla: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026.

[4] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. https://openreview.net/forum?id=WFYbBOEOtv.

[5] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

[6] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024.

[7] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[8] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[9] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[10] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[11] 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.

[12] Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

[13] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[14] Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, et al. Gr-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[15] Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19752–19763, 2025.

[16] Cheng Chi, Zhenjia Xu, Brian Ichter, and Avinash Shah. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[17] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[18] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 130:33–55, 2022.

[19] Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[20] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36: 9156–9172, 2023.

[21] Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, brian ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Pack Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning. In *The Twelfth International Conference on Learning Representations*, 2024. https://openreview.net/forum?id=9pKtcJcMP3.

[22] Yao Feng, Hengkai Tan, Xinyi Mao, Chendong Xiang, Guodong Liu, Shuhe Huang, Hang Su, and Jun Zhu. Vidar: Embodied video diffusion model for generalist manipulation. *arXiv preprint arXiv:2507.12898*, 2025.

[23] Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025.
[24] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Sharan Narang. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, 2024.
[25] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022.
[26] Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37: 112386–112410, 2024.
[27] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.
[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.
[29] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
[30] Shibo Hao, Sainbayar Gu, Haotian Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.
[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.
[32] Yucheng Hu et al. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.
[33] Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. Ladi-wm: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.
[34] Zixuan Huang, Xiang Li, Zhaoyang Lv, and James M Rehg. How much 3d do video foundation models encode? *arXiv preprint arXiv:2512.19949*, 2025.
[35] Zixuan Huang, Xiang Li, Zhaoyang Lv, and James M Rehg. How much 3d do video foundation models encode? *arXiv preprint arXiv:2512.19949*, 2025.
[36] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
[37] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.
[38] Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yi-An Ma, and Lianhui Qin. Ladir: Latent diffusion enhances llms for text reasoning. *arXiv preprint arXiv:2510.04573*, 2025.
[39] Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Yi-An Ma, and Lianhui Qin. Beyond mode elicitation: Diversity-preserving reinforcement learning via latent diffusion reasoner. *arXiv preprint arXiv:2602.01705*, 2026.
[40] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
[41] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
[42] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.
[43] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
[44] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.
[45] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.
[46] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

[47] Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, et al. Genie envisioner: A unified world foundation platform for robotic manipulation. *arXiv preprint arXiv:2508.05635*, 2025.
[48] Yaron Lipman, Ricky Chen, Heli Ben-Hamu, Max Nickel, and Manzil Zaheer Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
[49] Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, Zhengping Che, Jian Tang, Pheng-Ann Heng, and Shanghang Zhang. Last_0: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. *arXiv preprint arXiv:2601.05248*, 2026.
[50] Jinghui Lu, Jiayi Guan, Zhijian Huang, Jinlong Li, Guang Li, Lingdong Kong, Yingyan Li, Han Wang, Shaoqing Xu, Yuechen Luo, et al. Onevl: One-step latent reasoning and planning with vision-language explanation. *arXiv preprint arXiv:2604.18486*, 2026.
[51] Calvin Luo, Zilai Zeng, Yilun Du, and Chen Sun. Solving new tasks by adapting internet video knowledge. *arXiv preprint arXiv:2504.15369*, 2025.
[52] Yuechen Luo, Fang Li, Shaoqing Xu, Yang Ji, Zehan Zhang, Bing Wang, Yuannan Shen, Jianwei Cui, Long Chen, Guang Chen, Hangjun Ye, Zhi-Xin Yang, and Fuxi Wen. Last-vla: Thinking in latent spatio-temporal space for vision-language-action in autonomous driving. *arXiv preprint arXiv:2603.01928*, 2026.
[53] Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.
[54] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.
[55] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *International Conference on Learning Representations*, 2025.
[56] Nikolay Nikolov, Giuliano Albanese, Sombit Dey, Aleksandar Yanev, Luc Van Gool, Jan-Nico Zaech, and Danda Pani Paudel. Spear-1: Scaling beyond robot demonstrations via 3d understanding. *arXiv preprint arXiv:2511.17411*, 2025.
[57] OpenAI. Introducing gpt-5. https://openai.com/index/introducing-gpt-5-for-developers/, 2025.
[58] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
[59] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.
[60] 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.
[61] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.
[62] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
[63] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.
[64] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.
[65] Chenguo Shang et al. Worldarena: Benchmarking embodied video generation models as world simulators. *arXiv preprint arXiv:2602.08971*, 2026.
[66] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8075–8084, 2021.
[67] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
[68] GigaBrain Team, Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Haoyun Li, Jie Li, Jiagang Zhu, Lv Feng, et al. Gigabrain-0: A world model-powered vision-language-action model. *arXiv preprint arXiv:2510.19430*, 2025.
[69] MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

[70] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
[71] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. https://qwen.ai/blog?id=qwen3.5.
[72] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.
[73] Junming Wang, Teng Pu, Wingmun Fung, Jindong Wang, Shanchang Wang, Yuan Deng, Shuyuan Wang, Ziwei Liu, Kunhao Pan, Ping Yang, et al. Xrzero-g0: Pushing the frontier of dexterous robotic manipulation with interfaces, quality and ratios. *arXiv preprint arXiv:2604.13001*, 2026.
[74] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.
[75] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.
[76] John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. Dual-stream diffusion for world-model augmented vision-language-action model. *arXiv preprint arXiv:2510.27607*, 2025.
[77] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.
[78] Xianjin Wu, Dingkang Liang, Tianrui Feng, Kui Xia, Yumeng Zhang, Xiaofan Li, Xiao Tan, and Xiang Bai. Generation models know space: Unleashing implicit 3d priors for scene understanding. *arXiv preprint arXiv:2603.19235*, 2026.
[79] Hongwei Xue, Tiankai Hang, Yanhong Zeng, Yuchong Sun, Bei Liu, Huan Yang, Jianlong Fu, and Baining Guo. Advancing high-resolution video-language representation with large-scale video transcriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5036–5045, 2022.
[80] An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. https://arxiv.org/abs/2412.15115.
[81] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pages 83048–83077, 2025.
[82] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pages 28213–28239, 2025.
[83] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.
[84] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. In *NeurIPS*, 2024.
[85] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.
[86] 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.
[87] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning (CoRL)*, 2024.
[88] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *Advances in Neural Information Processing Systems*, 38:24195–24228, 2026.
[89] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.
[90] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. https://arxiv.org/abs/2304.13705.
[91] Zangwei Zheng, Xiangyu Peng, Yuxuan Lou, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, et al. Open-sora 2.0: Training a commercial-level video generation model in $200k. *arXiv preprint arXiv:2503.09642*, 2025.
[92] Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. Dualcot-vla: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models. *arXiv preprint arXiv:2603.22280*, 2026.
[93] Pengfei Zhou, Liliang Chen, Shengcong Chen, Di Chen, Wenzhi Zhao, Rongjun Jin, Guanghui Ren, and Jianlan Luo. Act2goal: From world model to general goal-conditioned policy. *arXiv preprint arXiv:2512.23541*, 2025.

[94] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research (PMLR)*, pages 61885–61896, 2024.

[95] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[96] Haoyi Zhu, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Chunhua Shen, Jiangmiao Pang, and Tong He. Aether: Geometric-aware unified world modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8535–8546, 2025.

[97] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.