ActWorld: From Explorable to Interactive World Model via Action-Aware Memory

Zhexiao Xiong[1,2,*], **Yizhi Song**[2], **Hao Kang**[2], **Qing Yan**[2], **Liming Jiang**[2], **Jenson Yang**[2], **Zhoujie Fu**[2], **Stathi Fotiadis**[2], **Angtian Wang**[2], **Zichuan Liu**[2], **Bo Liu**[2], **Yiding Yang**[2], **Xin Lu**[2], **Nathan Jacobs**[1]

[1]Washington University in St. Louis, [2]Intelligent Creation, ByteDance

[*]Work done during internship at ByteDance

Abstract

Interactive world models aim to simulate environment dynamics under real-time user actions. However, their action vocabulary is largely confined to navigation: most actions correspond to motion (e.g., walk, turn, look around), while interaction with objects in the scene (e.g., pick up plates, open doors, or trigger physical responses) is either absent, restricted to game domains, or relegated to prompt-to-full-video scenarios. The resulting worlds are visually explorable but not truly actionable. In this work, we present **ActWorld**, an interactive world model that extends prior navigation-centric generators to support mid-rollout object interaction within a chunk-autoregressive framework. We argue that the navigation–interaction gap stems from two bottlenecks. First, a data bottleneck: the lack of human–object interaction data with accurate, dense labels. Second, a memory bottleneck: recency-biased history compression in existing world models discards the event-transition frames that causally determine subsequent object states, leading to an action-forgetting pathology. On the data side, we construct a **100K interaction video dataset**, each annotated with per-chunk captions via chain-of-thought reasoning. On the model side, we introduce a **hierarchical action-aware memory** design that routes history compression by interaction importance, complemented by a persistent memory bank that maintains event-update and object-identity tokens across long rollouts. Experiments show that ActWorld supports both flexible navigation and rich object interaction within a single model, substantially improving interaction fidelity over navigation-only baselines without sacrificing viewpoint control.

Date: June 15, 2026

1 Introduction

Building world models that can simulate environment dynamics and support real-time interaction has emerged as a central challenge in artificial intelligence, with broad implications for gaming, embodied AI, autonomous driving, and content creation. The long-term goal is to learn how the world evolves directly from visual experience and to allow agents, whether human or artificial, to act within the simulated environment through controllable inputs. Early efforts [13, 20] established the feasibility of neural world simulation with diffusion-based generators. Since then, advances has been made on long-horizon consistency [23, 25], real-time high-resolution generation [7, 18, 22, 23, 32], and cross-domain controllability [12, 19, 26].

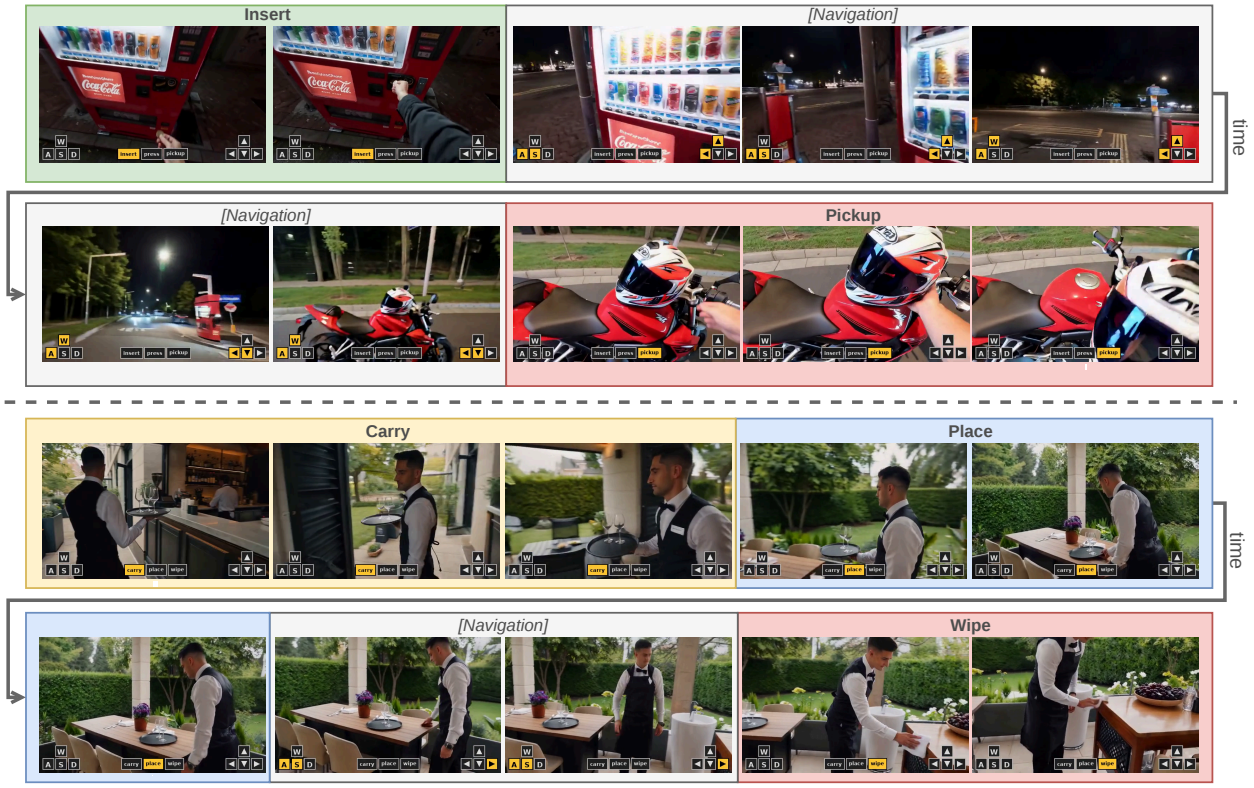

Figure 1 ActWorld is an interactive world model that handles both long-horizon navigation and mid-rollout object interaction within a single rollout, under per-frame keyboard and mouse control (WASD + arrow keys overlaid on each frame; yellow marks the active key). Each row is one continuous trajectory: time flows left-to-right and wraps into the next row (arrow t); colored bands separate navigation segments from object-interaction segments, with the action label shown above.

Despite this progress, a fundamental limitation remains: **existing interactive world models predominantly focus on locomotion and viewpoint control, while largely neglecting object-level interaction**. The majority of current systems [12, 18, 19, 23, 24] condition generation on keyboard or mouse inputs that drive movement and camera changes, producing worlds that are visually explorable but not truly actionable—agents can walk, turn, and navigate, but cannot pick up items, open doors, manipulate tools, or trigger meaningful physical responses from the environment. While PAN [26] supports language-specified manipulation commands, it operates in an offline, non-real-time regime unsuitable for interactive exploration. Domain-specific models such as Solaris [15] and Matrix-Game [4, 31] do support rudimentary object interactions (e.g., block breaking in Minecraft), they are confined to simple game environments and suffer from significant quality degradation during complex action sequences. More critically, no existing method jointly supports both fine-grained object interaction and flexible locomotion control within a unified real-time framework.

In this work, we present **ActWorld**, an interactive world model that unifies **rich object interaction** and **flexible viewpoint control** within a single real-time, chunk-autoregressive framework. We argue that the navigation–interaction gap stems from two compounding bottlenecks. The first is data: existing world-model datasets are overwhelmingly navigation-centric, providing little supervision for object-level dynamics. The second is memory design: current models organize history simply by temporal recency, preserving recent observations while compressing older ones. So frames where object states have been modified (e.g., a bottle being filled, a lamp turned on) are often heavily compressed when they are distant from the current video chunk. As a result, the model may lose the necessary evidence to predict subsequent states. We address both: on the data side, we collect a 100K interaction-dense video dataset with per-chunk dense captions from proprietary models; on the model side, we align memory granularity with interaction events, rather than time alone, so that causally critical frames survive compression regardless of their age. Our contributions are as follows:

- **Hierarchical action-aware memory**. To address the action-forgetting behavior, we introduce a novel memory mechanism: A local memory bank routes and amplifies interaction-critical frames within the sliding window; a persistent memory bank maintains compact event-update and object-identity tokens that survive beyond the window's eviction horizon.
- **An interaction-dense dataset and annotation pipeline**. Existing world-model datasets are largely navigation-centric, lacking both object-interaction coverage and fine-grained temporal labels. We construct a high-quality 100K-video dataset using proprietary video models, spanning 40 action categories, and annotate every chunk with a dense caption and an interaction-phase label.
- **A real-time interactive world model with navigation and object interaction**. We propose ActWorld, a real-time interactive world model that jointly supports flexible navigation and rich object interaction within a single framework. ActWorld integrates action-aware memory and the interaction data pipeline with a dual-branch camera conditioning module. Our model is validated on *I-Bench*, a new long-horizon benchmark that interleaves navigation and object interaction, where ActWorld substantially outperforms existing world models.

Experiments show that ActWorld supports flexible navigation and rich object interaction within a single real-time model. Compared with navigation-centric baselines, it substantially improves interaction fidelity while preserving locomotion and viewpoint controllability.

2 Related Works

Long Video Generation Most modern video diffusion models [8, 21] use bidirectional attention across all frames, precluding streaming generation. Causal autoregressive variants avoid this but suffer from exposure bias, which recent work closes through distillation or rollout-aware training [3, 6, 29, 33]. These methods target open-domain text-to-video synthesis and do not address action conditioning, spatial memory, or object-level interaction.

Real-Time Video Generation Even with causal architectures, multi-step sampling is the main throughput bottleneck. Few-step distillation [11, 17, 27–29] and pipeline-level optimizations [7, 23] together push generation to 24–40 FPS at 720p [18, 23]. These accelerations are designed for unconditional or text-conditioned generation; preserving fidelity under simultaneous locomotion and object-interaction conditioning remains underexplored.

Interactive World Models Interactive world models learn environment dynamics from data and simulate them in response to user actions. Early systems established feasibility in constrained, single-domain settings [2, 13, 20] but suffered from short horizons and fragile spatial memory. Recently, Infinite-World [25] and Matrix-Game 3.0 [23] extend coherent generation beyond 1,000 frames through memory compression and self-correction; WorldPlay [7, 18, 22] and Matrix-Game 3.0 reach real-time frame rates with high resolution; LingBot-World [19] and Yume-1.5 [12] generalize across diverse visual domains. However, action conditioning remains dominated by locomotion and viewpoint changes; object-level interaction is either absent, confined to narrow game manipulations [2, 4, 31], or relegated to offline language-conditioned generation [26]. **ActWorld** departs from this navigation-centric paradigm by jointly addressing the dataset scarcity and memory-design limitations that result in the navigation–interaction gap.

3 Method

ActWorld generates videos in a chunk-autoregressive manner, where each chunk is conditioned on past observations, user actions, and camera controls. Our method has three main components: per-chunk semantic captions for localized language conditioning (§3.1), decoupled camera control through Plücker-ray FiLM and symbolic text-camera channels (§3.2), and a hierarchical action-aware memory pipeline that re-routes interaction-relevant frames and persists event/object anchors across the latent buffer's eviction horizon (§3.3). A short distillation stage further reduces sampling cost for real-time inference (§3.4). We present the whole pipeline in Fig. 2.

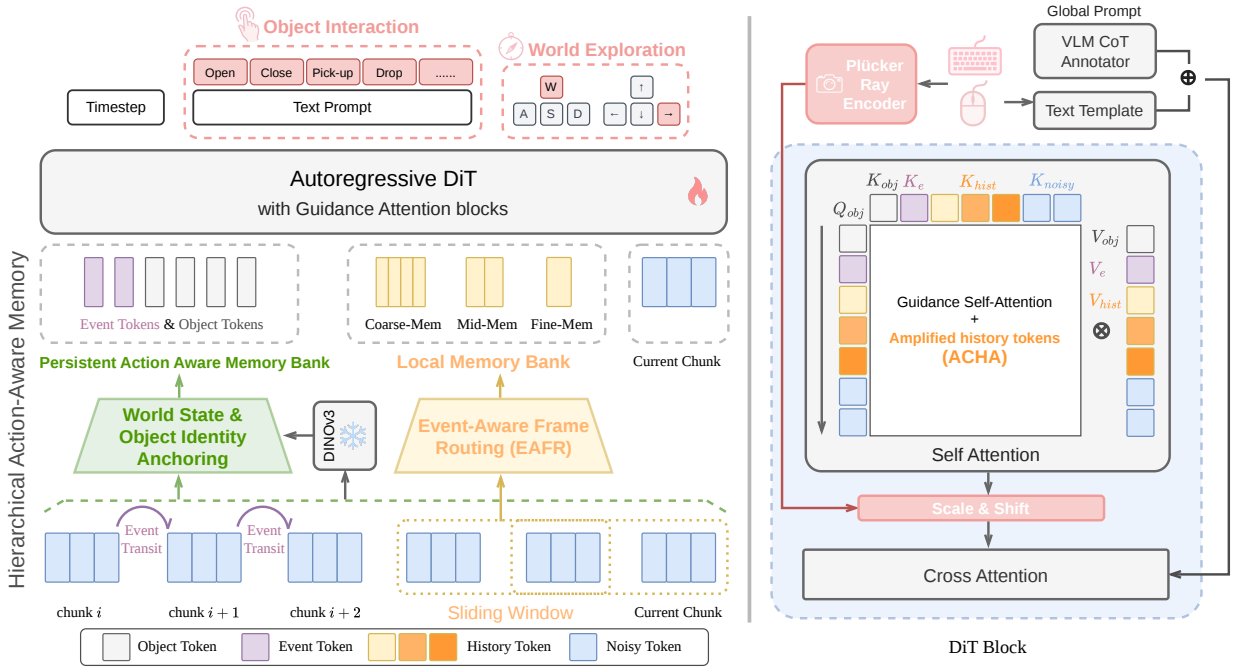

Figure 2 ActWorld pipeline. An autoregressive DiT generates each chunk under high-level object-interaction commands and low-level keyboard/mouse controls. **Left:** past observations flow through a hierarchical action-aware memory with two channels—a persistent action aware bank of event/object tokens that survives long navigation gaps, and a local bank of EAFR-routed coarse/mid/fine history tokens. **Right:** inside each DiT block, self-attention amplifies history keys via ACHA, Plücker rays drive a per-token scale-and-shift, and cross-attention ingests the per-chunk caption combined with a symbolic text-camera embedding.

3.1 Chain-of-Thought Per-Chunk Annotation

A single video-level caption is too coarse for chunk-autoregressive generation. Since the same caption is shared by multiple visually distinct chunks, the text condition does not specify which interaction phase the current chunk should depict, leading to temporally ambiguous conditioning and repetitive content. We instead annotate each chunk offline with a dedicated description and the structured labels consumed by §3.3. Every video is divided into non-overlapping 33-frame segments (≈1.4 s at 24 fps, aligned with the VAE latent window). For each chunk we extract 5 evenly-spaced keyframes and query a vision-language model (GPT-5.4). A direct prompt ("describe this chunk") yields hallucinated interactions, generic "scene continues" fallbacks, or confusion between camera motion and object interaction; we find that chain-of-thought (CoT) prompting substantially improves quality by forcing the VLM to reason over explicit visual evidence before committing to its outputs. The prompt instructs the model to:

- compare consecutive keyframe pairs and enumerate observable changes (object displacements, new contacts such as a hand grasping an object or an object placed on a surface, state transitions), ignoring camera motion;
- synthesize these observations to decide whether active interaction is occurring (y_k^{int}) and classify its phase y_k^{ph} into one of six phases: *approaching*, *reaching*, *contact*, *manipulating*, *completing*, and *post-action*. These phases respectively denote movement toward the target, object-directed reach, first physical touch, sustained object motion, final release or settlement, and frames after the action has finished;
- emit a 1–2 sentence description grounded solely in the accumulated frame-level evidence.

To maintain temporal coherence, chunks within each video are annotated sequentially, with chunk $t-1$'s description provided as context when annotating chunk t. The per-chunk descriptions are encoded offline by the frozen UMT5 [1] text encoder into embeddings $\mathbf{e}_t^{\text{chunk}} \in \mathbb{R}^{L\times d}$ used as the cross-attention condition; the

structured labels $(y_k^{\text{int}}, y_k^{\text{ph}})$ are passed downstream to the memory components in §3.3.

3.2 Keyboard/Mouse Control Conditioning

We distinguish low-level keyboard/mouse controls (camera translations and rotations) from high-level actions handled in §3.3, and expose the former through two complementary branches.

Geometric branch. Per-frame camera trajectories are converted into a per-pixel Plücker-ray tensor and routed through a shared `PlückerFiLM` module, following [19]. Our only departure is to reuse the module weights across all transformer blocks (parameter overhead ≈1.2% of the 14B DiT). The module emits per-token, zero-initialised FiLM scale-and-shift signals applied only to current-chunk hidden states (history KV is untouched), so toggling the branch off on a pretrained checkpoint recovers the baseline exactly.

Symbolic text-camera branch. The Plücker branch encodes camera motion as continuous geometry, but supervision is given as discrete user commands (9×9=81 `(keyboard, mouse)` combinations such as `W+D` with ↑ + →). Following [12], we map each command to a short natural-language template, encode it once with the frozen UMT5 text encoder, cache the embedding, and concatenate it with the per-chunk caption (§3.1) on the cross-attention path. An independent dropout $p_{\text{cam-txt}}$=0.1 on this branch discourages shortcut learning and biases the model toward the geometric signal.

Plücker-ray construction, pose normalisation, full FiLM equations, the 81-entry command vocabulary, and ablations are in Appendix. C.

3.3 Hierarchical Action-Aware Memory for High-Level Interactions

The conditioning mechanisms introduced so far control what each chunk depicts (§3.1) and how the camera moves (§3.2), but neither shapes which past frames the model conditions on. This third axis becomes critical under interaction commands, where the next chunk's outcome often hinges on sparse *contact* or *manipulating* frames many steps back. We diagnose this memory-side gap below and address it without modifying the backbone.

Problem formulation: action-forgetting in recency-based memory. Our backbone organizes the history buffer $\mathcal{H} = \{h_1, \ldots, h_{t-1}\}$ via a Multi-Term Memory (MTM) scheme that partitions past chunks into three buckets $\mathcal{S}_{\text{short}}, \mathcal{S}_{\text{mid}}, \mathcal{S}_{\text{long}}$ (named after their temporal distance) by recency, patchified with progressively larger spatio-temporal kernels and concatenated as history keys/values for DiT self-attention. For roaming trajectories this is nearly optimal: the frames that constrain the next observation are overwhelmingly recent. Interactive generation breaks this assumption. The frames that causally determine the next observation under one of our 40 high-level action commands (e.g., `pickup`, `open`, `attach`; full list in Appendix. H) are not recent ones but the sparse *contact* and *manipulating* moments, which typically sit in $\mathcal{S}_{\text{mid}}$ or $\mathcal{S}_{\text{long}}$ where object pose and contact geometry are degraded most; worse, the buffer's finite reach evicts anything older than $|\mathcal{H}|$ chunks, so objects that briefly leave the frame are lost on re-visit. We call this the action-forgetting problem: it is a memory-design rather than capacity issue, because *contact* and *manipulating* frames remain coarsely patchified no matter how far back the buffer reaches or how large the backbone grows.

Our response keeps the backbone unchanged and adds two complementary stages: a memory-retrieval re-routing stage that re-shapes which past frames the existing buckets, and a persistent action-aware memory bank that survives the latent buffer's eviction horizon. Both share three per-chunk labels emitted once offline by the CoT annotator of §3.1—an interaction flag y_k^{int}, a phase y_k^{ph}, and a video-level action class a—and are designed so that toggling them off recovers the baseline bit-for-bit.

Event-aware frame re-assignment (EAFR). We replace MTM's time-based bucketing with an importance-ranked split. For each past chunk k we score

$$w_k \;=\; \lambda_\varphi \, \varphi\Big(y_k^{\text{int}},\, y_k^{\text{ph}}\Big) \;+\; \lambda_r \, \exp(-(t-k)/\tau)\,, \tag{1}$$

where φ is a fixed phase prior that peaks on *contact* and *manipulating* and decays elsewhere, $\lambda_\varphi, \lambda_r > 0$ are scalar mixing weights, and the second term retains a recency bias with timescale τ relative to the current chunk index t. The buckets are filled greedily by descending w_k under the backbone's original size budgets, so we hereafter refer to them as $\mathcal{S}_{\text{fine}}, \mathcal{S}_{\text{mid}}, \mathcal{S}_{\text{coarse}}$. A *contact* frame from many steps ago can sit in fine-grained $\mathcal{S}_{\text{fine}}$ while a purely navigational recent frame drops to $\mathcal{S}_{\text{coarse}}$. RoPE positions are preserved across re-assignment: a migrated frame keeps its original time index, so EAFR changes which compressor a frame sees without altering its temporal position.

Action-conditioned history amplification (ACHA). The backbone already exposes a learnable per-head amplifier $s \in \mathbb{R}^{n_h}$ applied to history keys in self-attention, identical by default for a reach-to-grasp clip and a walk-past clip. ACHA makes it action-conditioned:

$$\alpha(\mathbf{e}_a) \;=\; \text{softplus}\Big(s + W_2\,\sigma(W_1\,\mathbf{e}_a)\Big), \qquad K'_{\text{hist}} = \alpha(\mathbf{e}_a) \odot K_{\text{hist}}, \tag{2}$$

where $\mathbf{e}_a \in \mathbb{R}^d$ is a learned embedding of the action class a, σ is SiLU, and W_1, W_2 form a $d \to d_b \to n_h$ bottleneck (d_b=256). The last layer is zero-initialised, so α collapses to the baseline amplifier at step 0; during training the MLP learns to sharpen attention onto history keys causally relevant to the current action, without adding tokens or a new attention stream.

Persistent action-aware memory bank. EAFR and ACHA still operate on the pixel-space latent stream, which is an inefficient carrier for sparse symbolic facts ("an object just left the gripper") and is bounded by the latent buffer's eviction horizon (anything older than $|\mathcal{H}|$ chunks is gone). We add a small structured channel that survives both limitations: an action-aware memory bank of at most K_{tot} tokens (default K_{tot}=16), preserved across chunk boundaries under a FIFO policy with phase-driven pinning, and prepended to the DiT self-attention input. The bank carries two complementary kinds of tokens: event tokens that record what happened and when, and object tokens that record what the affected object looks like. Both share the bank's segment slot and FIFO policy, and differ only in what triggers them and what they encode.

Event tokens. These fire at phase transitions: boundary events derived from changes in y^{ph} between consecutive chunks, complementing the per-chunk phase label itself. A rule-based writer scans y^{ph} for three such transitions—ENTER-MANIP (onset of *manipulating*), ENTER-COMPLETE (onset of *completing*), and RELEASE (end of *manipulating*)—and emits one token per firing at chunk k:

$$\mathbf{e}^{\text{evt}}_k \;=\; E_\xi[\xi_k] + E_{\text{ph}}[y^{\text{ph}}_k] + E_a[a] + \text{AttnPool}(\mathbf{h}_k), \tag{3}$$

where ξ_k is the firing transition tag; $E_\xi, E_{\text{ph}}, E_a$ are learned embedding tables for the transition, phase, and action class; their labels reuse the same vocabulary as EAFR/ACHA and E_a shares parameters with ACHA's action embedding, so no new vocabulary is introduced. $\text{AttnPool}(\mathbf{h}_k)$ is a single-query attention pool over the triggering chunk's patchified latent tokens, providing a visual summary of what the model just generated at the transition. Recency is left to the bank's FIFO ordering rather than encoded explicitly, so these tokens index memory by transition-tag × phase.

Object tokens. These capture the visual identity of the touched region. Whenever a chunk has $y^{\text{int}}_k = 1$ or carries a phase in $\{\textit{completing}, \textit{post-action}\}$, we emit up to K_{pc} object-anchor tokens (default K_{pc}=3) drawn from a frozen DINOv3 encoder. Concretely, for each of K_{kf}=5 evenly-spaced keyframes of chunk k we run DINOv3 to obtain patch features, score every patch by its L_2 norm (a saliency proxy that suppresses uniform-background patches), and keep the top K_{pc} across the chunk. Each surviving patch $\mathbf{f}^{\text{dino}}_{k,j} \in \mathbb{R}^{768}$ becomes a token:

$$\mathbf{e}^{\text{obj}}_{k,j} \;=\; W_v\big(\mathbf{f}^{\text{dino}}_{k,j}\big) + E_{\text{ph}}[y^{\text{ph}}_k] + E_a[a], \tag{4}$$

where W_v is a two-layer MLP with a zero-initialised final layer that lifts the patch feature into the transformer width d; this zero-init means the bank emits the zero vector at step 0, so toggling the writer on a pretrained checkpoint preserves the baseline forward pass exactly until learning kicks in. The phase and action embeddings E_{ph}, E_a are reused from event tokens; what makes object tokens distinct is the $W_v(\mathbf{f}^{\text{dino}})$ term, which encodes the visual signature of the touched region rather than a transition tag. As with event tokens, recency is left

to FIFO ordering, so these tokens index memory by visual signature × phase. Because DINOv3 features are view-stable, an anchor written at chunk k remains a useful query target after the camera turns away and back, letting the bank carry an object identity across long navigation gaps.

Bank operation and inference cost. Tokens enter a FIFO deque of capacity K_{tot}=16, except those whose source chunk has $y^{\text{ph}} \in \{\textit{contact}, \textit{manipulating}, \textit{completing}\}$, which are pinned for the rollout, preserving interaction moments through long stretches of navigation. The bank is prepended to the DiT input and enters self-attention only. At inference, object anchors are computed online by VAE-decoding each generated chunk and running a frozen DINOv3 [16] forward; gating this on interaction-related chunks keeps the average overhead below 15%. Bank mechanics (RoPE handling, segment embedding, training-time anchor pre-baking) are deferred to the appendix.

3.4 Few-Step Distillation for Long-Horizon Action-Conditioned Generation

For interactive use—each chunk must land in a fraction of a second—we follow the Stage-1 chunk-AR training above with two further stages from the Helios [30] acceleration recipe: (i) a multi-resolution flow-matching stage that splits denoising into K=3 resolution levels and regresses on the linear-flow velocity $\mathbf{v}^k = \mathbf{x}^k - \text{Upsample}(\mathbf{x}^{k-1})$, processing fewer tokens at coarser levels; and (ii) an adversarial DMD-style distillation that reduces the 50-step teacher to a 3-step generator G_θ. All conditioning streams from §3.1–3.3 ride through both stages as DiT inputs unchanged; full equations and hyperparameters follow [30].

I2V ODE warm-up. Our one deviation from [30] is the ODE-pair warm-up. Rather than t2v pairs, we construct pairs in an image-to-video regime: each clip's first ground-truth chunk is fixed as a clean conditioning image, and the teacher runs the multi-step sampler only on subsequent chunks under the same action labels and Plücker rays the student will later see. This matches the streaming-i2v interface at deployment (the user always provides a starting frame), removes regression variance from an extra t2v denoising step, and converges noticeably faster than t2v pairs of comparable size.

After distillation each 33-frame chunk is produced in three sampling steps without classifier-free guidance; the action-text dropout from §3.2 is kept on throughout Stage-3 so the CFG-free student still reacts to changing keyboard/mouse inputs. Full distillation hyperparameters are in Appendix. E.

4 Experiments

4.1 Data Generation Pipeline

Our training data combines two complementary sources. (1) We synthesize 100K videos with proprietary models to provide dense human-motion supervision, comprising 55K first-person and 45K third-person clips spanning 40 action categories; the third-person split covers both human-centric and non-human-centric actions. (2) We incorporate 400 hours of real-world data from a public dataset, mainly consisting of egocentric walking sequences in natural environments together with game footage. Detailed statistics of the data composition are provided in the supplementary material.

4.2 I-Bench: Long-Horizon Action--Navigation Benchmark

Existing benchmarks isolate either roaming through passive scenes (e.g., Yume, Sekai, VBench-Long) or scripted object interactions with fixed camera (WorldModelBench, VideoPhy-2); neither stresses what an interactive world model faces at deployment: chaining interactions while translating the viewpoint. We introduce **I-Bench**, in which every clip executes a long-horizon **(action sequence, camera sequence)** composite script.

I-Bench contains 300 prompts evenly split between first- and third-person views, organised as 30 semantically coherent sequences of 10 prompts each. Each prompt composes three action verbs from a 40-verb vocabulary interleaved with 2–3 camera primitives (translations, pans, tilts) described only in natural language, so the model must recover both signals from the prompt. Clips span 10 chunks of 33 frames. Each clip is

Table 1 VBench perceptual and consistency metrics on I-Bench. SC: subject-consistency, BC: background-consistency, MS: motion-smoothness, AQ: aesthetic-quality, IQ: imaging-quality, DD: dynamic-degree, TF: temporal-flickering, OC: overall-consistency, i2v-S / i2v-B: VBench-i2v subject / background. All scores are higher-is-better; bold marks the column best.

Method	SC	BC	MS	AQ	IQ	DD	TF	OC	i2v-S	i2v-B
Yume 1.5 [12]	0.743	0.872	0.985	0.432	0.645	1.000	0.962	0.180	0.930	0.943
HY-World 1.5 [7]	0.856	0.873	0.993	0.449	0.734	0.933	**0.977**	0.199	0.952	0.953
Lingbot-World [19]	0.724	0.866	0.979	0.436	0.693	0.983	0.966	0.182	0.917	0.935
Matrix-Game 3 [23]	0.662	0.854	0.981	0.377	0.681	1.000	0.954	0.169	0.862	0.889
Astra [34]	0.603	0.841	0.946	0.290	0.465	0.817	0.946	0.152	0.759	0.827
Infinite-World [25]	0.801	0.863	0.987	0.442	**0.748**	0.967	0.959	0.087	0.716	0.743
Ours	**0.871**	**0.896**	**0.991**	**0.485**	0.731	**1.000**	0.973	**0.201**	**0.954**	**0.957**

Table 2 VLM-AJ (VLM-Action-Judge) on I-Bench. IF: mean instruction-following score (0–3); Succ.: success rate (Level 3); ≥ 2: partial-or-better rate (Level $\geq$ 2). Higher is better.

Method	IF ↑	Succ. ↑	≥ 2 ↑
Yume 1.5 [12]	1.638	20.12	46.75
HY-World 1.5 [7]	0.709	5.19	18.70
Lingbot-World [19]	1.635	19.89	49.91
Matrix-Game 3 [23]	0.295	1.88	8.27
Astra [34]	0.949	5.83	19.36
Infinite-World [25]	0.237	1.51	3.95
Ours	**2.557**	**57.8**	**84.5**

Table 3 KMF (Key-Mouse-Following) on I-Bench. Acc_{full}: joint (keys, mouse) match; Acc_{keys} / $\text{Acc}_{\text{mouse}}$: per-axis accuracy. Computed from VIPE-recovered SE(3) trajectories.

Method	Acc_{full} ↑	Acc_{keys} ↑	$\text{Acc}_{\text{mouse}}$ ↑
Yume 1.5 [12]	4.82	31.79	15.71
HY-World 1.5 [7]	9.17	42.78	25.56
Lingbot-World [19]	2.67	28.00	13.33
Matrix-Game 3 [23]	20.00	**45.01**	40.83
Astra [34]	11.43	28.57	28.57
Infinite-World [25]	3.00	24.50	15.00
Ours	**20.62**	41.02	**43.67**

annotated with a global caption, per-chunk phase and sub-action descriptions, and a VIPE-recovered camera trajectory [5]; details in the supplement.

4.3 Evaluation Protocol

We evaluate every ablation along three complementary axes: VBench captures visual quality and temporal consistency, VLM-AJ (VLM-Action-Judge) measures semantic instruction following, and KMF (Key-Mouse-Following) measures geometric controllability of the generated camera trajectory; a faithful interactive world model has to win on all three. All pipelines run on the I-Bench testset introduced in Sec. 4.2.

(i) VBench perceptual and consistency suite. We adopt the same VBench-1.0 and VBench-i2v dimensions used by recent video world-model papers [12, 19, 25], computed via their native VBench entry points without modification.

(ii) VLM-AJ: VLM-Action-Judge. VBench is blind to whether the scripted action actually happens; we therefore adopt the four-level instruction-following rubric of WorldModelBench [9], also used by WorldSimBench [14]. For each chunk a judge VLM is shown four uniformly sampled frames together with the per-chunk GT description and rates action completion on a 0–3 scale (0: action absent; 1: wrong motion; 2: attempted but not completed; 3: fully completed). We report mean score, success rate (Level 3), and partial-or-better rate (Level $\geq$ 2), plus a per-action-class breakdown.

(iii) KMF: Key-Mouse-Following. KMF probes geometric controllability through a closed loop: starting from a ground-truth (keys, mouse) instruction sequence, we generate a video, recover its trajectory, and check whether each chunk moves in the commanded direction. Concretely, given a generated clip we (a) run VIPE [5], the same SLAM-style monocular pose estimator used to label the training corpus, to extract a per-frame SE(3) trajectory, with frames where SLAM fails recovered by nearest-valid forward/backward fill; (b) partition the trajectory into the same number of chunks as the GT instruction sequence and uniformly resample each to a 33-frame window, so that baselines with different native chunk granularities are compared at the same temporal scale; and (c) map each chunk's pose change to a discrete (keys, mouse) label via simple geometric

Figure 3 Qualitative visualization of **ActWorld** rollouts across diverse scenes that require both free-form navigation and object-centric interaction within a single continuous trajectory. Each row shows temporally ordered frames with per-frame keyboard/mouse controls overlaid on the images (active inputs highlighted in yellow), illustrating that the model preserves viewpoint controllability while producing coherent interaction outcomes.

rules. A predicted chunk is counted as correct only if both its keyboard and mouse labels match the ground truth; the per-video KMF score is the fraction of correct chunks, averaged over all videos in I-Bench. We report joint accuracy Acc_{full} together with keys-only and mouse-only accuracies.

4.4 Results

We show qualitative results in Fig. 3. We also compare ActWorld against six representative interactive / world-model video generators: Yume 1.5 [12], HY-World 1.5 [7], Lingbot-World [19], Matrix-Game 3 [23], Astra [34], and Infinite-World [25]. All baselines are evaluated on I-Bench under their official released checkpoints. Tables 1–3 report the three evaluation axes of Sec. 4.3, and Figures 1, 4 provide qualitative comparisons.

Quantitative. Across the three evaluation axes (Tables 1–3), ActWorld achieves the best or near-best score on each, with the largest margin on semantic instruction following (Table 2, where our Level-3 success rate of 57.8% more than doubles every baseline). This confirms that the hierarchical action-aware memory addresses the action-forgetting pathology while preserving the visual quality and viewpoint control of navigation-only baselines.

Qualitative. Figure 1 shows ActWorld executing multi-action sequences within a single continuous trajectory under WASD/arrow control, including *Insert* → *Pickup* and *Carry* → *Place* → *Wipe*. More results can be found in Appendix. F. Figure 4 contrasts a *Pour* → *Stir* sequence in this mixed navigation-and-interaction setting. Existing baselines cannot faithfully follow the fine-grained interaction steps: Yume 1.5 and HY-World 1.5 lose track of the manipulated objects partway through, and Lingbot-World drifts off the scene entirely. This is a systematic limitation of navigation-centric world models, whereas ActWorld preserves both the manipulated objects and the commanded action ordering across the full clip. Together, these results

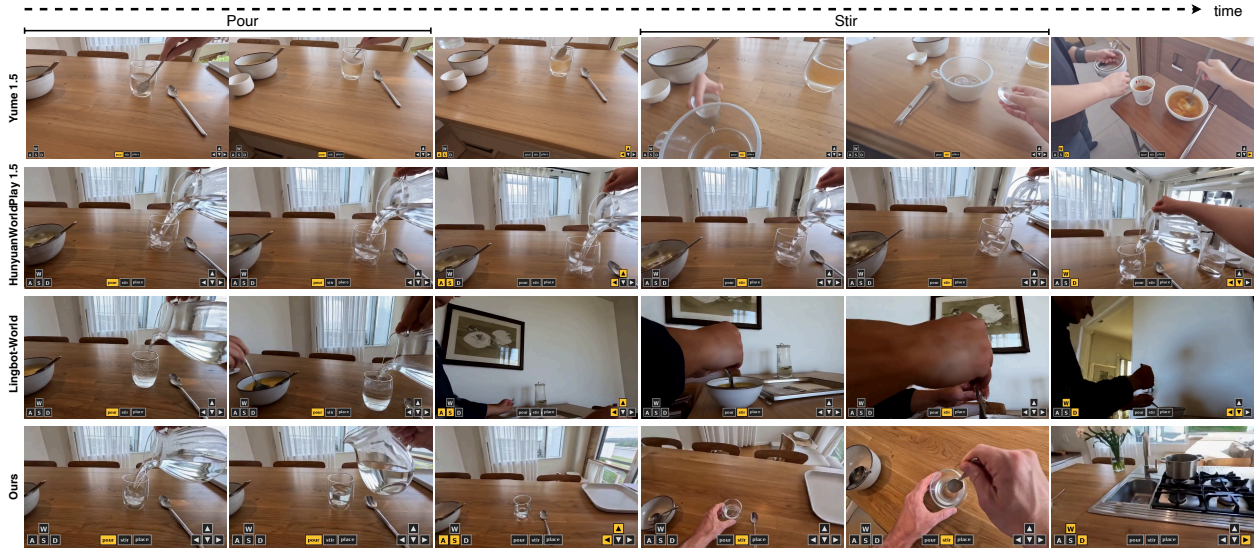

Figure 4 Qualitative comparison on a long-horizon, multi-step interaction sequence. Existing models often fail to do fine-grained human-object interaction or the full manipulation sequence. By contrast, **ActWorld** preserves scene coherence and follows the commanded interaction sequence more faithfully across time.

Table 4 Component ablation on I-Bench. Columns are grouped by evaluation axis: VBench (perceptual/consistency dims; DD/TF/OC dropped due to saturation or low differentiability), VLM-AJ (semantic instruction following), and KMF (geometric controllability). All rows include the Plücker FiLM camera conditioner and per-chunk prompt substitution as baseline components. EAFR: importance-ranked history split; ACHA: action-conditioned history-key amplifier; EventMem: phase-transition tokens.

Run	VBench							VLM-AJ			KMF		
	SC	BC	MS	AQ	IQ	i2v-S	i2v-B	IF	Succ.	≥ 2	full	keys	mouse
CP+Plücker(base)	0.803	0.895	0.985	0.479	0.714	0.934	0.942	2.326	52.9	80.3	20.29	40.65	43.11
+EAFR	0.834	0.894	0.987	0.482	0.718	0.930	0.942	2.345	53.8	82.1	20.45	40.72	43.22
+EAFR+ACHA	0.844	0.894	0.985	0.484	0.709	0.928	0.941	2.413	54.0	82.8	20.58	40.85	43.35
+EventMem, full	**0.871**	**0.896**	**0.991**	**0.485**	**0.731**	**0.954**	**0.957**	**2.557**	**57.8**	**84.5**	**20.62**	**41.02**	**43.67**

validate that navigation and fine-grained object interaction can be unified within a single real-time rollout.

4.5 Ablation Study

Table 4 ablates the three memory components on I-Bench-mini. Adding EAFR (re-routing contact and manipulation frames into fine-grained memory) gives a foundational lift; adding ACHA sharpens semantic instruction-following further; Event Memory delivers the largest single jump (subject consistency 0.844 $\rightarrow$ 0.871, Level-3 success 54.0% $\rightarrow$ 57.8%), confirming that the persistent event and object slots do the heavy lifting on object identity tracking across long rollouts.

4.6 User Study

We conduct a user study against the same six baselines on three 1–5 criteria: action following, key/mouse following (overlaid WASD keys and mouse arrows), and overall quality (clarity, temporal consistency, visual artifacts). Tab. 5 shows ActWorld ranks first on all three, with the widest margin on action following.

Table 5 Results of user study. Our method outperforms all baselines across all three criteria.

	Astra	HY-W1.5	LingBot	MG-3	Yume	**Ours**
Action Follow.	1.35	1.89	2.68	1.18	2.58	**4.05**
Key/Mouse Follow.	1.35	2.35	2.23	1.73	2.22	**3.69**
Overall Quality	1.31	3.15	2.62	1.70	3.02	**3.92**

5 Conclusion

We presented **ActWorld**, an interactive world model that extends prior navigation-centric video generators to support mid-rollout object interaction within a single real-time framework. Our central argument is that the navigation–interaction gap is primarily a memory-design issue rather than an architectural one: recency-based history compression systematically discards the very frames that drive object-level dynamics. Our memory and conditioning innovations address this mismatch and produce a model that performs object interaction while preserving keyboard-mouse viewpoint control. We view this as a step toward truly interactive world models, with real-time, AI-generated gameplay as the most immediate downstream application, and embodied planning and co-driven content creation following naturally once a learned simulator can both move and act.

References

[1] Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. arXiv preprint arXiv:2304.09151, 2023.

[2] Decart and Etched. Oasis: A universe in a transformer. Technical report / project page, October 2024. URL https://about.decart.ai/publications/oasis-interactive-ai-video-game-model. Interactive world model for real-time AI-generated gameplay.

[3] Yuwei Guo, Ceyuan Yang, Hao He, Yang Zhao, Meng Wei, Zhenheng Yang, Weilin Huang, and Dahua Lin. End-to-end training for autoregressive video diffusion via self-resampling. arXiv preprint arXiv:2512.15702, 2025.

[4] Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, et al. Matrix-game 2.0: An open-source real-time and streaming interactive world model. arXiv preprint arXiv:2508.13009, 2025.

[5] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, Jiawei Ren, Kevin Xie, Joydeep Biswas, Laura Leal-Taixe, and Sanja Fidler. Vipe: Video pose engine for 3d geometric perception. In NVIDIA Research Whitepapers arXiv:2508.10934, 2025.

[6] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. arXiv preprint arXiv:2506.08009, 2025.

[7] Team HunyuanWorld. Hy-world 1.5: A systematic framework for interactive world modeling with real-time latency and geometric consistency. arXiv preprint, 2025.

[8] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. arXiv preprint arXiv:2412.03603, 2024.

[9] Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph E. Gonzalez, Ion Stoica, Song Han, and Yao Lu. Worldmodelbench: Judging video generation models as world models. In Advances in Neural Information Processing Systems, volume 38, 2025.

[10] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. arXiv preprint arXiv:2501.08316, 2025.

[11] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick Von Platen, ApolinÃĄrio Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. arXiv preprint arXiv:2311.05556, 2023.

[12] Xiaofeng Mao, Zhen Li, Chuanhao Li, Xiaojie Xu, Kaining Ying, Tong He, Jiangmiao Pang, Yu Qiao, and Kaipeng Zhang. Yume-1.5: A text-controlled interactive world generation model. arXiv preprint arXiv:2512.22096, 2025.

[13] Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model. Google DeepMind Blog, 2024. URL https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/.

[14] Yiran Qin, Zhelun Shi, Jiwen Yu, Xijun Wang, Enshen Zhou, Lijun Li, Zhenfei Yin, Xihui Liu, Lu Sheng, Jing Shao, et al. Worldsimbench: Towards video generation models as world simulators. arXiv preprint arXiv:2410.18072, 2024.

[15] Georgy Savva, Oscar Michel, Daohan Lu, Suppakit Waiwitlikhit, Timothy Meehan, Dhairya Mishra, Srivats Poddar, Jack Lu, and Saining Xie. Solaris: Building a multiplayer video world model in minecraft. arXiv preprint arXiv:2602.22208, 2026.

[16] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. arXiv preprint arXiv:2508.10104, 2025.

[17] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Proceedings of the 40th International Conference on Machine Learning, 2023.

[18] Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. arXiv preprint arXiv:2512.14614, 2025.

[19] Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing open-source world models. arXiv preprint arXiv:2601.20540, 2026.

[20] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. arXiv preprint arXiv:2408.14837, 2024.

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

[22] Zehan Wang, Tengfei Wang, Haiyu Zhang, Xuhui Zuo, Junta Wu, Haoyuan Wang, Wenqiang Sun, Zhenwei Wang, Chenjie Cao, Hengshuang Zhao, et al. Worldcompass: Reinforcement learning for long-horizon world models. arXiv preprint, 2026.

[23] Zile Wang, Zexiang Liu, Jaixing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, et al. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. arXiv preprint arXiv:2604.08995, 2026.

[24] Ruiqi Wu, Xuanhua He, Meng Cheng, Tianyu Yang, Yong Zhang, Zhuoliang Kang, Xunliang Cai, Xiaoming Wei, Chunle Guo, Chongyi Li, and Ming-Ming Cheng. Infinite-world: Scaling interactive world models to 1000-frame horizons via pose-free hierarchical memory. arXiv preprint arXiv:2602.02393, 2026.

[25] Ruiqi Wu, Xuanhua He, Meng Cheng, Tianyu Yang, Yong Zhang, Zhuoliang Kang, Xunliang Cai, Xiaoming Wei, Chunle Guo, Chongyi Li, et al. Infinite-world: Scaling interactive world models to 1000-frame horizons via pose-free hierarchical memory. arXiv preprint arXiv:2602.02393, 2026.

[26] Jiannan Xiang, Yi Gu, Zihan Liu, Zeyu Feng, Qiyue Gao, Yiyan Hu, Benhao Huang, Guangyi Liu, Yichi Yang, Kun Zhou, et al. Pan: A world model for general, interactable, and long-horizon world simulation. arXiv preprint arXiv:2511.09057, 2025.

[27] 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.

[28] 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.

[29] Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In CVPR, 2025.

[30] Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. arXiv preprint arXiv:2603.04379, 2026.

[31] Yifan Zhang, Chunli Peng, Boyang Wang, Puyi Wang, Qingcheng Zhu, Fei Kang, Biao Jiang, Zedong Gao, Eric Li, Yang Liu, and Yahui Zhou. Matrix-game: Interactive world foundation model. arXiv preprint arXiv:2506.18701, 2025.

[32] Hongxiang Zhao, Xingchen Liu, Mutian Xu, Yiming Hao, Weikai Chen, and Xiaoguang Han. Taste-rob: Advancing video generation of task-oriented hand-object interaction for generalizable robotic manipulation. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 27683–27693, 2025.

[33] Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. arXiv preprint arXiv:2602.02214, 2026.

[34] Yixuan Zhu, Jiaqi Feng, Wenzhao Zheng, Yuan Gao, Xin Tao, Pengfei Wan, Jie Zhou, and Jiwen Lu. Astra: General interactive world model with autoregressive denoising. arXiv preprint arXiv:2512.08931, 2025.

Appendix

A Data Generation Pipeline

This section details the offline data preparation pipeline summarised in Fig. 5. All stages are run once per video and cached to disk, so no VLM, VAE, or DINOv3 forward pass is required during training; per-step training cost is dominated by the diffusion target alone.

Chunking and visual encoding. Each raw 24 fps video is split into non-overlapping segments of 33 consecutive frames (≈1.4 s), aligned with the VAE's latent temporal stride. Each chunk is encoded by a frozen VAE into a latent tensor that serves as the diffusion target during training and the autoregressive output during rollout. Chunks for which any of the 33 source frames is invalid (scene cut or missing frame) are dropped together with their downstream annotations.

Camera condition. Per-frame camera-to-world poses $(\mathbf{R}, \mathbf{t})$ and intrinsics $\mathbf{K}$ are recovered with the VIPE pose estimator [5]. From $(\mathbf{R}, \mathbf{t}, \mathbf{K})$ we build a per-pixel Plücker-ray map and downsample it to the latent spatial grid; this map drives the PlückerFiLM modulation of §3.2 (full construction in §C). Poses are normalised to the chunk's first frame so the conditioning is translation-invariant within the chunk.

Per-chunk chain-of-thought annotation. For each chunk we extract five evenly-spaced keyframes ($t \in \{0.0, 0.3, 0.7, 1.0, 1.3\}$ s) and query a frozen vision-language model (GPT-5.4) under a chain-of-thought prompt that has access to (i) the five keyframes, (ii) the dataset-supplied video-level action label, and (iii) the description of the previous chunk for temporal continuity. The reasoning template walks the model through five steps:

1. compute pairwise frame differences and summarise the dominant motion;
2. determine whether subject and object are in physical contact;
3. check whether the supplied action label is consistent with the observed motion (otherwise emit `ACTION_MISMATCH`);
4. assign an interaction-phase tag $y_k^{\text{ph}} \in \mathcal{P}$ from the taxonomy of §3.1;
5. write a 1–2 sentence semantic description grounded solely in the per-frame evidence and ignoring camera motion.

The structured output schema is fixed: `HAS_INTERACTION` ∈ {yes, no} (the interaction flag y_k^{int}), `ACTION_MISMATCH` ∈ {yes, no}, `PHASE` $\in \mathcal{P}$ (the phase label y_k^{ph}), plus a free-form `DESCRIPTION` string. Together with the video-level action class $a \in \mathcal{A}$ from the dataset manifest (Appendix. H), these form the per-chunk label triple $(y_k^{\text{int}}, y_k^{\text{ph}}, a)$ consumed by the three memory modules of §3.3.

Object anchors via DINOv3. For each of the same five keyframes we run a frozen DINOv3-B/16 encoder on a 224×224 centre crop and retain the 32 patches with the largest L_2 feature norm per keyframe as candidate object anchors. To keep the saliency distribution consistent with inference, where the bank writer only sees model-generated content, the DINOv3 input frames are decoded from the chunk's VAE latent rather than read from raw RGB; this train–test alignment removes the small but non-trivial domain gap introduced by VAE compression. The pre-baked top-32 patches form the candidate pool from which the action-aware memory bank's writer selects at training time (full wiring in §D).

Final cached training sample. Each cached sample bundles eight fields:

- `vae_latent`: VAE-encoded 33-frame chunk (diffusion target);
- `camera_poses`, `intrinsics`: per-frame $(\mathbf{R}, \mathbf{t})$ and $\mathbf{K}$;

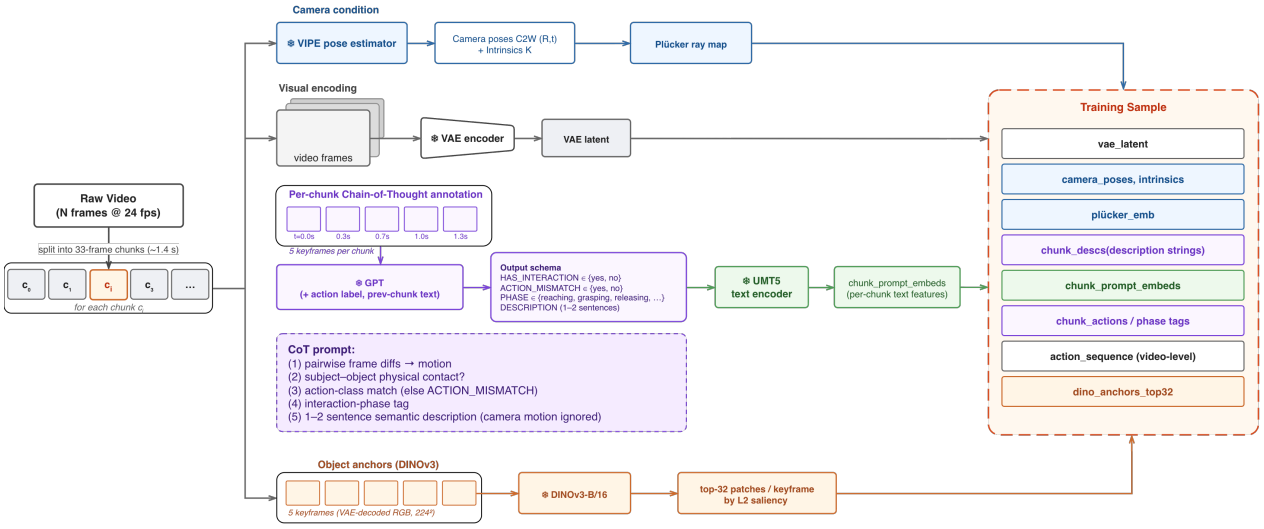

Figure 5 Data generation pipeline. Each raw 24 fps video is split into non-overlapping 33-frame chunks (≈1.4 s). Per chunk, four offline stages run in parallel: VAE-latent extraction (diffusion target); Plücker rays from VIPE-recovered camera poses (FiLM condition for §3.2); chain-of-thought annotation over five evenly-spaced keyframes via a frozen VLM (GPT-5.4), producing an interaction flag, an action-mismatch flag, a phase label $y_k^{\text{ph}} \in \mathcal{P}$, and a 1–2 sentence description encoded by frozen UMT5; and DINOv3-B/16 object anchors retaining the top-32 saliency-ranked patches per keyframe (candidate pool for the bank writer of §3.3). All eight fields form the final cached training sample (right).

- `plucker_emb`: pre-rendered Plücker-ray map at latent resolution;
- `chunk_descs`: raw per-chunk description string (diagnostics only);
- `chunk_prompt_embeds`: UMT5 embedding of the description ($\mathbf{e}_t^{\text{chunk}}$);
- `chunk_actions`: structured labels ($y_k^{\text{int}}, y_k^{\text{ph}}$, `ACTION_MISMATCH`) per chunk;
- `action_sequence`: video-level action class a from the 40-verb vocabulary (Appendix. H);
- `dino_anchors_top32`: pre-baked DINOv3 top-32 patches per keyframe (5 keyframes × 32 patches).

All fields are written to disk once per dataset; the runtime data loader concatenates them into the model-side inputs without further processing.

B Label Handling and Zero-Initialisation Invariants

Fall-back on unannotated clips. Clips that lack one or more of the per-chunk labels (y_k^{int}, y_k^{ph}, a) defined in §A (e.g., pure navigation data scraped without an interaction track) emit `None` for the missing fields. EAFR's frame importance score defaults to its recency component only when y^{ph} is absent, ACHA's MLP receives a zero action embedding when a is absent, and the action-aware memory bank's writer gate fires only when the relevant labels are present. In all three cases the per-sample fallback is bit-for-bit equivalent to the baseline, so mixed batches of annotated and unannotated clips train without any distributional shift.

B.1 Zero-initialisation Invariants

A subtle pitfall when adding modules to a pretrained backbone is that the mere presence of a new module at step 0 can shift the forward pass—through a non-zero output projection, a RoPE re-indexing, or a moved token—making any downstream metric change ambiguous between "the module learned something useful" and "the model recovered from a worse initialisation". Our design neutralises this on every new path:

- **EAFR.** Frames are re-assigned across short/mid/long buckets but their RoPE indices follow the original time stamps, so self-attention sees the same temporal embedding it would in the time-only baseline; the

only thing that changes is which compression kernel each frame meets.

- **ACHA.** The bottleneck MLP $W_2\,\sigma(W_1\,\mathbf{e}_a)$ that produces the action-conditioned delta on top of the per-head amplifier s has its final layer zero-initialised, so $\alpha(\mathbf{e}_a)=\mathrm{softplus}(s)$ at step 0, recovering the unconditioned amplifier.
- **Plücker FiLM.** The scale and shift heads W_s, W_b in (7) are zero-initialised; combined with frame-0 pose normalisation this guarantees $\widetilde{\mathbf{X}}^{(\ell)}_{\mathrm{cur}} = \mathbf{X}^{(\ell)}_{\mathrm{cur}}$ at step 0, so attaching the branch to a trained checkpoint does not disturb the existing forward pass.
- **Action-aware memory bank.** The bank's segment embedding, the event token's out-projection, and the object token's final visual projection W_v are all zero-initialised, so every prepended bank token contributes exactly $\mathbf{0}$ to the self-attention sum until trained.

Together these invariants ensure that toggling any subset of the new modules on a pretrained checkpoint produces a step-0 loss curve that matches the baseline to machine precision; every subsequent metric change is therefore attributable to learning, not to a re-init shock.

C Keyboard/Mouse Control Conditioning: implementation details

This subsection expands on the geometric Plücker branch and the symbolic text-camera branch summarised in §3.2.

Plücker-ray construction. Each training clip ships with a per-frame camera-to-world pose $\mathbf{P}_t \in \mathrm{SE}(3)$ and intrinsics $\mathbf{K}=[f_x, f_y, c_x, c_y]$ recovered by VIPE [5]. We normalise translations relative to frame 0 (rotations are unchanged), then downsample to one pose per latent frame (T_p=33 video frames $\to$ T_l=9 latent frames). For each latent frame t and each pixel (u, v) on the latent grid, we cast a ray from the camera origin and form the 6-D Plücker representation

$$\boldsymbol{\rho}_{t,u,v} \;=\; \big[\,\mathbf{r}_o(t)\times\mathbf{r}_d(t,u,v),\;\mathbf{r}_d(t,u,v)\,\big] \in \mathbb{R}^6, \tag{5}$$

where $\mathbf{r}_o(t)$ is the world-space camera origin at frame t and $\mathbf{r}_d(t,u,v)$ is the world-space ray direction for pixel (u,v) at frame t. This yields a tensor $\boldsymbol{\rho}\in\mathbb{R}^{B\times 6\times T_l\times H_p\times W_p}$ aligned one-to-one with the latent grid.

*Patch packing (*Pack*).* The DiT operates on latent patch tokens. The packing operator Pack rearranges the 6-D Plücker rays inside each patch into a single feature vector, producing one packed block per latent patch-token. The subsequent linear W_{patch} projects each packed block into the transformer width d.

PlückerFiLM module. Given $\boldsymbol{\rho}$, the module first produces a per-patch cam token via a residual MLP:

$$\mathbf{z}^{\mathrm{cam}} \;=\; W_2\,\sigma\big(W_1\,\mathbf{e}\big) + \mathbf{e}, \qquad \mathbf{e} \;=\; W_{\mathrm{patch}}\big(\mathrm{Pack}(\boldsymbol{\rho})\big), \tag{6}$$

where σ is SiLU. At every transformer block ℓ, the same $\mathbf{z}^{\mathrm{cam}}$ drives a per-token FiLM scale-and-shift modulation applied only to the current-chunk hidden states $\mathbf{X}^{(\ell)}_{\mathrm{cur}}$ (history tokens from the KV cache are untouched):

$$\begin{aligned} \mathbf{c} &= W_4\,\sigma\big(W_3\,\mathbf{z}^{\mathrm{cam}}\big) + \mathbf{z}^{\mathrm{cam}}, \\ \widetilde{\mathbf{X}}^{(\ell)}_{\mathrm{cur}} &= \big(\mathbf{1}+\mathbf{s}\big)\odot\mathbf{X}^{(\ell)}_{\mathrm{cur}} + \mathbf{b}, \quad \mathbf{s}=W_s\,\mathbf{c},\;\; \mathbf{b}=W_b\,\mathbf{c}. \end{aligned} \tag{7}$$

The scale/shift heads W_s, W_b are zero-initialised; combined with the frame-0 pose normalisation, this gives a step-0 identity $\widetilde{\mathbf{X}}^{(\ell)}_{\mathrm{cur}} = \mathbf{X}^{(\ell)}_{\mathrm{cur}}$, so enabling the branch on a pretrained checkpoint preserves the baseline forward pass exactly. The same module weights $\{W_1,\ldots,W_4,W_s,W_b\}$ are reused at every transformer block, contributing $\approx$1.2% of the 14 B DiT.

Per-token vs. broadcast modulation. The FiLM signal in Eq. 7 is per-token rather than broadcast over space, since yaw/pitch rotations and translations move different image regions differently—e.g., a camera yaw shifts the right side of the image faster than the left in pixel velocity. A single global control vector cannot express this anisotropy.

Symbolic command vocabulary. The dataset is annotated with 9×9=81 `(keyboard, mouse)` combinations: 9 keyboard categories {*idle*, W, A, S, D, W+A, W+D, S+A, S+D} crossed with 9 mouse categories {*idle*, ↑, ↓, ←, →, ↑→, ↑←, ↓→, ↓←}. Each combination is mapped to a short natural-language template of the form "Person moves ⟨keyboard direction⟩. Camera tilts ⟨mouse direction⟩." Examples:

- `(idle, idle)`: "Person stays still. Camera holds steady."
- `(W, ↑→)`: "Person moves forward. Camera tilts up and turns right."
- `(W+D, ↑→)`: "Person moves forward and right. Camera tilts up and turns right."

Cross-attention conditioning. Each command template is encoded once by the frozen UMT5 text encoder, producing an embedding $\mathbf{e}^{\text{cam-txt}} \in \mathbb{R}^{L_a \times d_t}$ that we cache so neither training nor inference pays a text-encoder cost. At training time the per-chunk action label indexes into this cache; the embedding is concatenated with the per-chunk caption embedding from §3.1 along the sequence dimension before the DiT cross-attention:

$$\mathbf{h}_{\text{enc}} \;=\; \left[\, \mathbf{e}_t^{\text{chunk}} ;\; \mathbf{e}_t^{\text{cam-txt}} \,\right] \;\in\; \mathbb{R}^{(L+L_a)\times d_t}. \tag{8}$$

Because the symbolic label is a tempting shortcut—one short phrase shared across a whole bucket of trajectories—we apply an independent dropout $p_{\text{cam-txt}}$=0.1 to $\mathbf{e}_t^{\text{cam-txt}}$ before concatenation, leaving $\mathbf{e}_t^{\text{chunk}}$ and the geometric Plücker tensor $\boldsymbol{\rho}$ intact. This biases the model toward the geometric branch and, empirically, drives the zero-initialised FiLM heads off the identity manifold.

D Action-aware Memory Bank: implementation details

Hyperparameters. We use K_{tot}=16 for total bank capacity, K_{kf}=5 keyframes per chunk for the object writer's DINOv3 forward, and K_{pc}=3 object-anchor tokens kept per qualifying chunk. The event writer fires at most once per chunk per transition tag ($\leq$ 3 events / chunk in practice). The object writer's auxiliary trigger phases (used in addition to $y_k^{\text{int}}=1$) are {*completing*, *post-action*}.

Bank insertion and attention path. At every chunk-autoregressive step, the bank's contents are prepended to the DiT input sequence ahead of the current-chunk noise tokens. Each slot carries a learnable segment embedding with one extra bit distinguishing event from object tokens, so the DiT layers can specialise to token type without recovering it from content alone. Bank tokens are written with all RoPE positional indices set to zero, so they participate in attention purely through their content embeddings without competing with the spatial RoPE grid on current-chunk patches. They enter the DiT self-attention only and do not modify the cross-attention path, which continues to carry the per-chunk semantic embedding $\mathbf{e}_t^{\text{chunk}}$ from §3.1. With K_{tot} bank tokens and current-chunk latent token counts on the order of $F{\cdot}H{\cdot}W$ (thousands), the bank adds well under 0.5% to the DiT input length and a comparable fraction to per-step compute.

FIFO and pinning. The bank is a fixed-capacity deque of size K_{tot}; when full, the oldest unpinned token is evicted. Tokens from chunks labelled *contact*, *manipulating*, or *completing* are pinned for the remainder of the rollout, while tokens from *approaching*, *reaching*, or *post-action* chunks recycle normally.

DINOv3 backbone. We use the publicly released DINOv3-B/16 (ViT-B with 14×14 patch tokens at 224^2 input, 768-dim outputs). Inputs are bilinearly resized from the chunk's native 384×640 keyframes and normalised with the standard ImageNet mean/std. The L_2 norm of each patch feature serves as the saliency score; we flatten the $K_{\text{kf}}\times P$ patches into a single pool and take the top K_{pc} across the chunk so a salient object caught in two consecutive keyframes contributes both views to the bank instead of being deduplicated.

Offline pre-baking. For every training video we run DINOv3 once over its keyframes and cache the top-32 patch features and saliency scores per chunk (`dino_anchors_top32`, ≈ 50 KB / chunk in fp16) into the feature `.pt` file. The training-time writer reads the cache and selects the top K_{pc} per chunk; this pushes the DINOv3 forward off the training critical path entirely. Pre-baking the full $\sim$100 K-clip set takes $\sim$2.5 h on 8×H100 (one rank per video shard).

Online inference. At inference the model produces brand-new chunks that have no pre-baked features. The pipeline therefore (i) VAE-decodes the just-generated chunk, (ii) samples K_{kf} evenly-spaced keyframes from the decoded 33-frame block, (iii) runs a single forward through frozen DINOv3 to obtain patch features, and (iv) admits the top patches under the same writer gate ($y_k^{\text{int}} = 1$ or $y_k^{\text{ph}} \in \{\textit{completing}, \textit{post-action}\}$). Pure-navigation chunks skip the decode + DINOv3 pass entirely. In practice $\sim$30–40 % of chunks fire the writer; the average inference overhead is below 15 % of the chunk-generation budget. Event tokens require no online computation: they operate purely on the symbolic y^{ph} stream the model already conditions on, and the embedding lookup $E_\xi[\xi_k] + E_{\text{ph}}[y_k^{\text{ph}}] + E_a[a]$ is constant-time.

E Distillation Details

This section provides full technical details of the distillation pipeline summarised in §3.4. The recipe closely follows Helios [30]; we restate the relevant equations here for self-containment.

Stage 2: Multi-resolution flow matching. We split denoising into K=3 resolution levels, partition $[0, 1000]$ into boundaries T_0=1000>T_1>T_2>T_3=0, and at each level k train along the linear flow path with ground-truth velocity $\mathbf{v}^k = \mathbf{x}^k - \text{Upsample}(\mathbf{x}^{k-1})$,

$$\mathcal{L}_{\text{flow}} \;=\; \mathbb{E}\big[\, \| u_\theta^k(\mathbf{x}_t^k, \mathbf{y}, \lambda_t, k) - \mathbf{v}^k \|_2^2 \,\big], \tag{9}$$

where $\mathbf{y}$ bundles the chunk caption, the text-camera embedding, the action label, and the Plücker tensor. At inference we allocate (N_1, N_2, N_3) steps across the three levels and bridge transitions by nearest-neighbour upsampling followed by re-noising; coarser levels process fewer tokens, so the per-chunk cost drops substantially.

Stage 3: Backward simulation under pure teacher forcing. We distil the 50-step model into a 3-step generator G_θ with a DMD-style [27, 28] objective tailored to the autoregressive chunk setting. Training uses pure teacher forcing: only real history latents are fed, and we generate a single chunk per step rather than rolling out long sequences. Backward simulation is performed in K levels matching the flow-matching pyramid; at level k we estimate

$$\mathbf{x}_0^k \;=\; \mathbf{x}_t^k - \lambda_t \, u_\theta^k(\mathbf{x}_t^k, \mathbf{y}, \lambda_t, k), \tag{10}$$

reconstruct $\mathbf{x}_t^k$ along the flow path, iterate to convergence, and pass $\mathbf{x}_0^k$ to level k+1. The I2V ODE warm-up used to initialize G_θ before Stage 3 begins is described in §3.4 of the main paper and is the only deviation from [30].

Stage 3 objectives. Distribution matching uses a CFG'd real score p_{real} and a fake score p_{fake}. To escape the teacher's expressive ceiling we attach multi-granularity discriminator heads D to selected DiT layers of p_{fake} and train them on real data with a non-saturated GAN loss plus an approximate-R_1 regulariser [10]. The Stage-3 objectives are

$$\mathcal{L}_{G_\theta} = \mathcal{L}_{\text{DMD}} + w_G \, \mathcal{L}_G, \tag{11}$$

$$\mathcal{L}_{p_{\text{fake}}} = \mathcal{L}_{\text{Flow}} + w_D \, \mathcal{L}_D, \tag{12}$$

and G_θ is updated once per several p_{fake} updates following the TTUR scheduling of [29].

F More Visualizations

We show further visualization in Fig. 6.

G Training Details

Initialisation. We initialise from a publicly released Helios [30] 14 B chunk-autoregressive checkpoint, itself adapted from the Wan2.1-14B [21] bidirectional text-to-video diffusion model pretrained on open-domain data. All architectural additions of §3.2–3.3 (Plücker FiLM, ACHA bottleneck, memory-bank embedding tables, object-token MLP W_v) have zero-initialised final layers, so at step 0 the modified checkpoint is bit-for-bit equivalent to Helios.

Task. Stage-1 training is run in an image-to-video (I2V) regime: for each clip the first ground-truth chunk is fixed as a clean conditioning image, and the model is supervised to denoise subsequent chunks under the per-chunk caption (§3.1), Plücker rays and symbolic camera command (§3.2), and action-aware memory channel (§3.3). The same I2V interface is preserved at inference, so the deployed model always conditions on a user-provided starting frame.

Hardware and schedule. Stage-1 trains for 5,000 optimisation steps on 48×H100 GPUs with bf16 mixed precision. Stage-2 multi-resolution flow-matching warm-up trains for another 5,000 steps on the same hardware. Stage-3 distillation comprises a 3,000-step I2V ODE warm-up followed by 3,000 steps of GAN distillation. Both stages follow the Helios [30] schedule; full equations and the I2V ODE-pair construction are deferred to Appendix. E.

H High-Level Action Vocabulary

Our training corpus is annotated with 40 high-level action commands selected to span both human-centric manipulation and dynamic locomotion. We organise them into eight semantic groups:

- **Pick-and-place (8):** `pickup`, `putdown`, `place`, `lift`, `drop`, `hold`, `grab`, `carry`.
- **Open/close & lock (4):** `open`, `close`, `lock`, `unlock`.
- **Attach/detach (6):** `attach`, `detach`, `insert`, `remove`, `plug`, `unplug`.
- **Force application & locomotion (6):** `push`, `pull`, `drag`, `throw`, `kick`, `drive`.
- **Surface and tool manipulation (6):** `wipe`, `swipe`, `stir`, `pour`, `peel`, `cut`.
- **Discrete contact (3):** `tap`, `press`, `switch`.
- **Handover (3):** `give`, `receive`, `release`.
- **Orientation & material (4):** `rotate`, `slide`, `fold`, `pack`.

The corpus is partitioned across three sources by viewpoint and subject: a 50K-clip first-person ego-centric subset spanning all 40 categories (1,250 clips each); a 30K-clip third-person human-manipulation subset covering the 32 manipulation-focused categories (∼938 each); and a 10K-clip third-person non-human subset (animals, vehicles) restricted to the 8 dynamic-motion categories `carry`, `drag`, `drive`, `drop`, `grab`, `hold`, `pull`, `release` (1,250 each), totaling 90,000 annotated clips.

Figure 6 Further results generated by ActWorld.