EA-WM: Event-Aware Generative World Model with Structured Kinematic-to-Visual Action Fields

Zhaoyang Yang[1] **Yurun Jin**[3,4]
Kai Chen[2,3,5*] **Cong Huang**[2,3*] **Lizhe Qi**[1*]

[1]Fudan University [2]Zhongguancun Academy
[3]Zhongguancun Institute of Artificial Intelligence
[4]University of Science and Technology of China [5]DeepCybo

https://github.com/Shownx-c/EA-WM

Abstract

Pretrained video diffusion models provide powerful spatiotemporal generative priors, making them a natural foundation for robotic world models. While recent world-action models jointly optimize future videos and actions, they predominantly treat video generation as an auxiliary representation for policy learning. Consequently, they insufficiently explore the inverse problem: leveraging action signals to guide video synthesis, thereby often failing to preserve precise robot spatial geometry and fine-grained robot-object interaction dynamics in the generated rollouts. To bridge this gap, we present **EA-WM**, an Event-Aware Generative World Model that effectively closes the loop between kinematic control and visual perception. Rather than injecting joint or end-effector actions as abstract, low-dimensional tokens, EA-WM projects actions and kinematic states directly into the target camera view as *Structured Kinematic-to-Visual Action Fields* (KVAFs). To fully exploit this geometrically grounded representation, we introduce event-aware bidirectional fusion blocks that modulate cross-branch attention, capturing object state changes and interaction dynamics. On WorldArena, EA-WM achieves an EWMScore of 60.38 across all 16 metrics, exceeding the strongest compared text-conditioned baseline by 2.47 points, and a P3CScore of 76.60.

1 Introduction

Recent video foundation models have rapidly improved the ability to synthesize temporally coherent, high-fidelity videos. From latent video diffusion models and space-time diffusion architectures to diffusion-transformer systems such as CogVideoX [32] and the open Wan series [28, 27, 8], these models learn strong priors over appearance, motion, and scene evolution from large-scale video corpora [4, 2, 32, 28]. This progress has motivated the view that scaled video generation can move beyond visual content creation toward world simulation [15, 1]. Since robot manipulation is fundamentally a process of predicting how actions change visual states over time, such spatiotemporal priors make video generation models a natural foundation for robotic world modeling.

Driven by this potential, recent robotics research has begun to adapt these video priors for several roles. Some methods [29, 19, 37] use video world models as predictive simulators to roll out the visual consequences of candidate actions, enabling action-conditioned video prediction, visual planning, model-based reinforcement learning, and fine-grained robot-object interaction simulation. Other work [9, 14, 18, 13, 25] uses generated rollouts as robot-learning infrastructure: they can synthesize additional robot trajectories beyond manual data collection, provide video-based supervision for

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

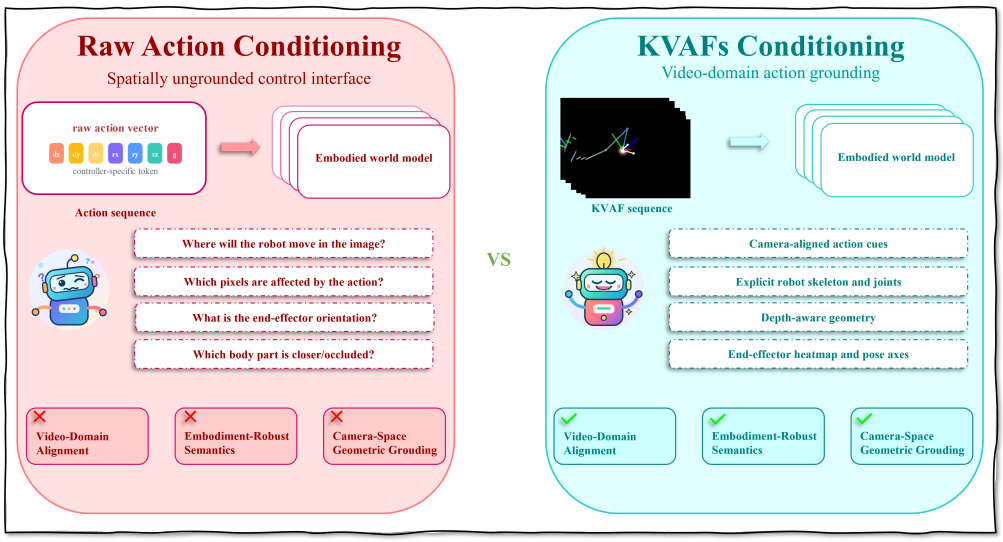

Figure 1: Comparison between direct low-dimensional action conditioning and the proposed *Structured Kinematic-to-Visual Action Fields* (KVAFs).

policy learning, and offer scalable proxies for evaluating VLA policies without repeatedly executing every policy on physical robots.

Meanwhile, recent world-action models [36, 33, 10] have begun to jointly tune video generation and action modeling within unified generative frameworks, showing that future videos can provide dense world representations for action generation, policy learning, planning, and value estimation. However, these works predominantly treat video generation merely as an auxiliary representation to optimize action prediction. Consequently, they insufficiently explore the crucial inverse problem: how to effectively leverage control signals to guide accurate video synthesis. This oversight leads to a critical limitation: the generated videos often fail to preserve the precise spatial geometry of robot motion and the complex dynamics of robot-object interactions. As high-quality video rollouts are increasingly relied upon as predictive simulators and synthetic data engines for downstream tasks [9, 14, 18, 13, 25], mitigating this degradation in rollout quality is imperative. A fundamental obstacle lies in the domain misalignment between low-dimensional control signals and high-dimensional video synthesis. Standard practices typically inject raw joint-space parameters, end-effector vectors, or abstract tokens as conditionings [29, 19, 37, 17]. While computationally compact, these representations are heavily tied to specific robot embodiments and carry limited spatial context [35]. They force the video generator to implicitly deduce cross-domain kinematics, frequently resulting in a failure to accurately render robot geometry or capture nuanced physical interactions. Furthermore, conventional architectures often lack an effective interaction mechanism between the action and video pathways, causing the generator to overlook fine-grained interaction information between the robot and objects.

To address these challenges, we propose **EA-WM**, an event-aware robotic video world model conditioned on *Structured Kinematic-to-Visual Action Fields* (KVAFs). At the representation level, EA-WM constructs KVAFs by lifting low-dimensional robot actions and kinematic states into the target camera view through robot kinematics and camera projection. Specifically, KVAFs render depth-aware arm structures, joint landmarks, gripper geometry, end-effector heatmaps, and pose cues as temporally aligned visual fields. Compared with raw action vectors, KVAFs bring action conditioning into the same image domain as future video generation, thereby alleviating domain misalignment and providing richer spatial and geometric cues for robotic world modeling, as shown in Figure 1. At the architectural level, EA-WM augments a diffusion-transformer backbone with a dedicated KVAF branch and interval-based bidirectional fusion, facilitating continuous interaction between structured kinematic cues and video features. Crucially, we introduce an event-aware bidirectional fusion mechanism driven by *Event-Difference Latent Supervision* (EDLS). By supervising event predictions with VAE-encoded temporal difference latents, EDLS compels the model to dynamically allocate attention not only to the robot's geometric progression but also to regions undergoing state

transitions and interaction dynamics, leading to significantly more physically consistent rollouts. The key contributions of this work are summarized as follows:

- We introduce **EA-WM**, an innovative Event-Aware Generative World Model framework for robotic video generation. By projecting low-dimensional kinematic states directly into the target camera view, our framework creates *Structured Kinematic-to-Visual Action Fields* (KVAFs), fundamentally resolving the domain misalignment between raw action tokens and video-domain generation.
- We propose an *event-aware bidirectional fusion* mechanism driven by *Event-Difference Latent Supervision* (EDLS). This module explicitly regularizes cross-stream attention using temporal difference latents, compelling the model to dynamically attend to critical regions of robot motion, object state transitions, and robot-object interaction dynamics, leading to more physically consistent robot videos.
- We achieve superior performance on the WorldArena benchmark. Compared to existing state-of-the-art baselines, EA-WM significantly enhances the physical adherence, 3D geometric accuracy, and fine-grained controllability of generated rollouts, establishing a new standard for robotic video modeling.

2 Related Work

2.1 Robotic world models based on video generation models

Recent diffusion-based video generators, including Stable Video Diffusion [4], Lumiere [2], CogVideoX, and the Wan series [32, 28, 27, 8], have substantially improved temporal coherence, motion realism, and scene-consistent synthesis, making them increasingly useful as priors for world simulation [15, 1]. These priors are especially useful in robotics, where world models must predict how actions change future visual states. Existing robotic video world models span action-conditioned simulators and forward models for planning, model-based RL, and interaction simulation [29, 19, 37], infrastructure methods that use generated videos as synthetic trajectories, supervision, or policy-learning proxies [9, 14], and learned video environments for scalable VLA evaluation and planning [18, 13, 25]. Recent world-action models further couple video and action generation in unified frameworks [36, 33, 10]. Despite these advances, how to fully exploit action information to improve future robot-video generation remains insufficiently explored.

2.2 World action models

World Action Models (WAMs) are unified generative models that predict both robot actions and future visual states in an aligned manner, as formulated in DreamZero [33]. Recent work [36, 12, 33, 22, 20, 24, 10, 3, 16] in this line mainly treats future video modeling as dense supervision or latent guidance for better action generation and policy learning. These WAM-style methods demonstrate that future videos can regularize action learning with physical dynamics, provide denser supervision than action-only imitation, and improve generalization for robot control. However, their emphasis remains largely on how video generation can improve action prediction or policy learning. The reverse direction—using action information as structured guidance to improve future robot-video generation—remains much less explored, even though rollout quality directly affects the usefulness of robotic world models as predictive simulators, planning substrates, and policy-evaluation environments [9, 14, 18, 13, 25].

A key question is therefore how action information should be represented. Existing methods have explored raw joint-space or end-effector vectors, 7-DoF control tokens, latent actions, universal action tokens, and spatial action abstractions. VideoVLA [22], UVA [12], Motus [3], and ViPRA [20] represent action information through numerical, latent, or video-conditioned action spaces, while UniAct [35] and SpatialVLA [17] study more transferable or spatially structured action representations. While compact and effective for policy learning, these representations remain weakly grounded in the target camera view and carry limited explicit spatial and geometric information for video generation. WoG [24] also highlights the challenge of preserving sufficient fine-grained information under compact future-conditioning representations. Recent visual representations lift actions into image or video space via RGB action targets [23], virtual robot renders [26], pixel-grounded multiview action videos [34], or multiview heatmap videos [11]. However, their action

channels are often intentionally compact and robot-centric: they prioritize joint targets, rendered robot bodies, arm-motion cues, or heatmaps, so fine-grained kinematic detail may still be limited, and robot-object interaction information is often modeled only indirectly through companion RGB branches or downstream video generation rather than encoded explicitly in the action representation itself.

Recent methods also project robot motion into visual space. EgoDemoGen combines rendered robot motion with reprojected scene videos for novel-view demonstration synthesis [31]; BridgeV2W uses pixel-aligned embodiment masks and a flow-based motion loss [5]; and Kinema4D uses kinematic pointmaps for joint RGB and 4D scene generation [30]. Our distinction lies not in projection alone, but in the articulated topology, depth, joint, gripper, and end-effector information retained by KVAFs, and in their joint generation with RGB through EDLS-guided bidirectional fusion. The controlled representation comparisons are reported in Sec. A.7.

3 Method

EA-WM is a robotic video world model built upon Wan2.2-TI2V [28, 27]. We study two separately trained configurations. **EA-WM-Joint** receives an initial RGB observation and a language instruction and jointly generates future RGB videos and KVAF sequences; ground-truth future robot trajectories are not provided at inference. **EA-WM-Cond** additionally receives reference KVAFs constructed from a robot trajectory and predicts future RGB videos. The overall pipeline is shown in Figure 2. The reference-conditioned variant in Table 4 removes the KVAF prediction head; the two configurations are detailed in Sec. A.8. Instead of injecting low-dimensional actions directly into the video generator, EA-WM first transforms actions and kinematic states into *Structured Kinematic-to-Visual Action Fields* (KVAFs), which provide camera-aligned visual cues of robot geometry, motion, and end-effector intent. Both RGB videos and KVAFs are encoded into the latent space of the pretrained video VAE, allowing action information to guide generation in the same representation space as the target videos. To effectively exploit this structured action signal, EA-WM introduces a dedicated KVAF branch and event-aware bidirectional fusion blocks on top of the Wan2.2 video DiT. The KVAF branch learns to accurately predict KVAFs, encouraging the branch to understand the geometric and motion information embedded in KVAFs. Meanwhile, the event-aware bidirectional fusion blocks are guided by *Event-Difference Latent Supervision* (EDLS), which uses VAE-encoded temporal difference latents to supervise event predictions, encouraging the model to capture object motion and robot-object interaction cues. Through cross-attention, the video backbone exchanges information with the KVAF branch, allowing structured action geometry and event-aware interaction cues to be shared with the generative stream for more accurate future video generation.

3.1 Structured Kinematic-to-Visual Action Fields

We introduce *Structured Kinematic-to-Visual Action Fields* (KVAFs), which convert low-dimensional robot actions and kinematic states into camera-aligned visual fields for future video generation. Given a trajectory of length T, we construct a sequence $\mathcal{V} = \{\mathbf{V}_t\}_{t=1}^{T}$, where each frame $\mathbf{V}_t \in \mathbb{R}^{H\times W\times 3}$ is rendered in the target camera view. Here H and W denote the image height and width, and t indexes the video timestep.

At time step t, we take arm joint values $\mathbf{q}_t$, gripper states g_t, end-effector poses $\boldsymbol{\xi}_t$, and camera parameters $(\mathbf{K}_t, \mathbf{E}_t)$ as input. Here $\mathbf{K}_t$ denotes the camera intrinsic matrix, and $\mathbf{E}_t$ denotes the world-to-camera extrinsic matrix. For simplicity, we omit the left/right arm index in the notation; in practice, KVAFs are constructed for both arms. We first recover 3D robot geometry by forward kinematics,

$$\mathbf{T}_k^W(t) = \mathbf{T}_{k-1}^W(t)\mathbf{T}_k^{\mathrm{orig}}\mathbf{T}_k^{\mathrm{mot}}(q_{t,k}), \tag{1}$$

where $\mathbf{T}_k^W(t)$ is the world-frame pose of the k-th robot link, $\mathbf{T}_k^{\mathrm{orig}}$ is the static joint transform from the robot kinematic chain, and $\mathbf{T}_k^{\mathrm{mot}}(q_{t,k})$ is the motion transform induced by the joint value $q_{t,k}$. The resulting link poses provide arm and gripper keypoints $\mathcal{P}_t = \{\mathbf{p}_k^W(t)\}$ in the world frame.

We then project these 3D keypoints into the camera plane by

$$\mathbf{p}^C = \mathbf{E}_t \begin{bmatrix} \mathbf{p}^W \\ 1 \end{bmatrix}, \qquad \hat{\mathbf{u}} = \mathbf{K}_t \mathbf{p}_{1:3}^C, \qquad (u, v) = \left(\frac{\hat{u}_x}{\hat{u}_z}, \frac{\hat{u}_y}{\hat{u}_z} \right). \tag{2}$$

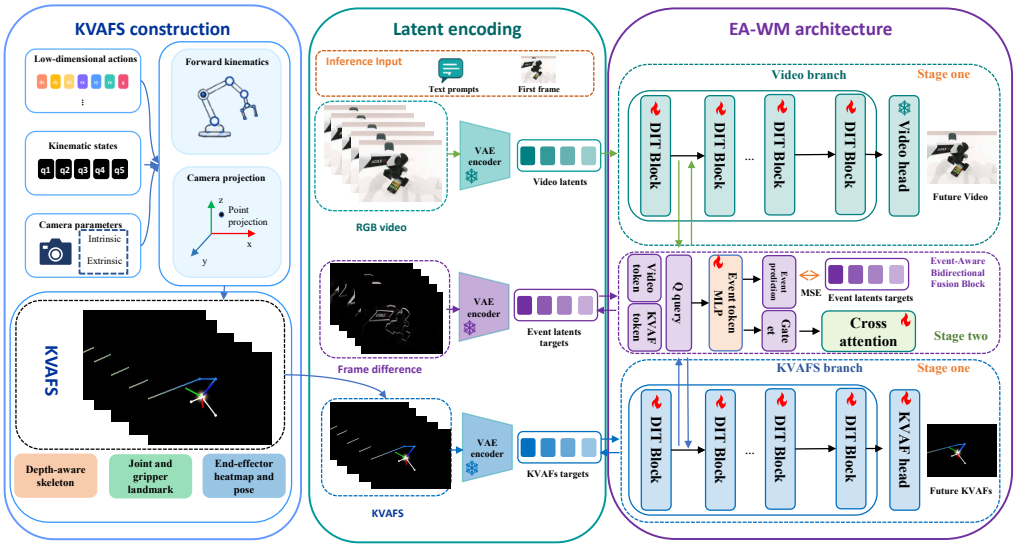

Figure 2: Overview of EA-WM. Robot actions and kinematic states are first lifted into camera-aligned KVAFs. RGB videos and KVAFs are encoded into the shared Wan2.2 latent space and processed by a video branch and a KVAF branch. Sparse event-aware bidirectional fusion exchanges information between the two branches, while EDLS guides the model to attend to motion and interaction changes.

Here $\mathbf{p}^W$ and $\mathbf{p}^C$ denote the same robot keypoint in the world and camera coordinate frames, respectively; $\hat{\mathbf{u}}$ is the homogeneous image coordinate, and (u, v) is the final projected pixel location. Points behind the camera are discarded.

Finally, we rasterize depth-aware arm skeletons, joint landmarks, gripper geometry, end-effector heatmaps, and pose axes on a black canvas to obtain

$$\mathbf{V}_t = \mathrm{Render}\big(\mathcal{P}_t, \boldsymbol{\xi}_t, \mathbf{K}_t, \mathbf{E}_t\big). \tag{3}$$

The rendering operator composes the projected robot keypoints, end-effector pose cues, and gripper-related visual elements into an RGB visual action field. By expressing robot actions in the same image domain as the target rollout, KVAFs provide spatially grounded and temporally aligned motion cues that are more compatible with video-based robotic world modeling than raw action tokens.

3.2 Event-Aware Generative World Model

EA-WM builds upon the Wan2.2-TI2V backbone and preserves its original text-conditioned video denoising path. To inject structured action information, we encode KVAFs with the same video VAE and introduce a dedicated KVAF branch in the latent space. After patchification, the noised video latent and KVAF latent are represented as two token streams, denoted by $\mathbf{H}^v$ and $\mathbf{H}^k$. The video stream follows the original Wan2.2 DiT blocks, while the KVAF stream is modeled by a full-depth copy of these blocks, allowing action information to remain as a structured visual stream rather than being compressed into a low-dimensional token.

We insert event-aware fusion modules at a sparse set of layers $\mathcal{S}$. For $\ell \in \mathcal{S}$, an event MLP first computes a shared event representation from the current video and KVAF tokens, and then predicts both a per-token event gate and an event latent:

$$\mathbf{M}_\ell = \Phi_\ell\left(\mathbf{H}^v_{\ell-1}, \mathbf{H}^k_\ell\right), \qquad \mathbf{G}_\ell = \Gamma_\ell(\mathbf{M}_\ell), \qquad \hat{\mathbf{E}}_\ell = \Psi_\ell(\mathbf{M}_\ell). \tag{4}$$

Here $\mathbf{G}_\ell$ is a per-token event gate that modulates cross-stream information exchange, while $\hat{\mathbf{E}}_\ell$ is the predicted event latent supervised by EDLS. Since both outputs are derived from the shared event representation $\mathbf{M}_\ell$, EDLS encourages $\mathbf{M}_\ell$ to encode temporal changes and interaction cues, which are then passed to the gate $\mathbf{G}_\ell$ and used to regulate video–KVAF fusion.

The event gate controls bidirectional cross-attention between the two streams:

$$\tilde{\mathbf{H}}^v_{\ell-1} = \mathbf{H}^v_{\ell-1} + \mathbf{G}_\ell \odot \mathrm{CA}_{v\leftarrow k}\left(\mathbf{H}^v_{\ell-1}, \mathbf{H}^k_\ell\right), \tag{5}$$

$$\tilde{\mathbf{H}}^k_{\ell} = \mathbf{H}^k_{\ell} + \mathbf{G}_\ell \odot \mathrm{CA}_{k\leftarrow v}\left(\mathbf{H}^k_{\ell}, \mathbf{H}^v_{\ell-1}\right). \tag{6}$$

The updated video tokens are then processed by the original Wan2.2 block, and the updated KVAF tokens are passed to the next KVAF branch block. At non-fusion layers, the two streams are updated without cross-stream fusion. In this way, EDLS does not merely add an auxiliary event prediction head; it shapes the shared event representation that produces the gate, and the gate directly controls how much KVAF information is injected into the video stream and how much scene information is fed back to the KVAF stream.

Our event terminology is inspired by event-based vision [7]. Unlike event cameras, which asynchronously record thresholded brightness changes, EDLS uses frame-synchronous absolute differences from standard RGB videos. These differences supervise the shared representation used to predict fusion gates during training and are not required at inference. To construct EDLS, we compute a frame-difference video from the input sequence $\{I_\tau\}_{\tau=1}^{T}$:

$$\Delta I_1 = \mathbf{0}, \qquad \Delta I_\tau = |I_\tau - I_{\tau-1}|, \quad \tau = 2, \ldots, T, \tag{7}$$

and encode it with the same VAE to obtain the event latent target $\mathbf{E}$. Let $\hat{\mathbf{Y}}^v$ and $\hat{\mathbf{Y}}^k$ be the predictions of the video and KVAF heads, and $\mathbf{Y}^v$ and $\mathbf{Y}^k$ be their corresponding flow-matching targets. The total training objective is

$$\mathcal{L} = \omega(t)\left(\left\|\hat{\mathbf{Y}}^v - \mathbf{Y}^v\right\|_2^2 + \left\|\hat{\mathbf{Y}}^k - \mathbf{Y}^k\right\|_2^2\right) + \lambda_{\mathrm{evt}} \frac{1}{|\mathcal{S}|} \sum_{\ell\in\mathcal{S}} \left\|\mathrm{Unpatchify}(\hat{\mathbf{E}}_\ell) - \mathbf{E}\right\|_2^2. \tag{8}$$

Here $\omega(t)$ denotes the timestep-dependent flow-matching training weight used by the scheduler, and λ_{evt} controls the strength of EDLS. The first video frame is treated as the image condition and excluded from the video prediction loss. This objective encourages EA-WM to use KVAF information in regions where motion and interaction changes occur, preserving action-consistent robot geometry while remaining sensitive to object motion and robot-object interactions.

4 Experiments

4.1 Experimental Details

We train EA-WM in two stages: the first stage freezes the event-aware fusion modules and trains the main DiT LoRA, KVAF-branch LoRA, and KVAF head to stabilize the two streams; the second stage unfreezes the fusion modules to learn cross-stream event-aware interaction. We evaluate generated videos with WorldArena [21], a unified benchmark for embodied world models. We use a LoRA rank of 32, and set the learning rate to 8×10^{-5} for all trainable components. Training is conducted on 32 H100 GPUs with a batch size of 32. Following WorldArena, we use the same RoboTwin dataset and segmentation method and evaluate all 16 metrics. We additionally report P3CScore as a task-focused summary of Physics Adherence, 3D Accuracy, and Controllability.

4.2 Experimental Results

Quantitative Analysis. Following WorldArena, we report EWMScore, the average of all 16 normalized metrics multiplied by 100. We also retain P3CScore as a task-focused average of the six robot-centric metrics shown in Table 1. All individual metrics are normalized to $[0, 1]$ and higher is better. The complete metric breakdown is provided in Tables 6 and 7.

EA-WM achieves an EWMScore of 60.38, outperforming CogVideoX, the strongest compared text-conditioned baseline, by 2.47 points across all 16 metrics. Its P3CScore is 76.60, corresponding to a 5.52-point improvement on the six robot-centric metrics. The improvement is particularly clear on metrics related to action-induced motion and interaction: EA-WM improves Interaction Quality from 0.594 to 0.682, Trajectory Accuracy from 0.353 to 0.430, and Instruction Following from 0.727 to 0.792. These gains suggest that introducing KVAFs and event-aware bidirectional fusion blocks helps the model better preserve robot-object interaction, follow commanded motion trajectories, and generate videos that are more consistent with the task instruction. EA-WM also improves Depth

Table 1: WorldArena comparison with text-conditioned baselines. All individual metrics are normalized to $[0, 1]$ and higher is better. EWMScore averages all 16 metrics, whereas P3CScore averages the six robot-centric metrics shown here; both aggregates are multiplied by 100. External baseline values are from WorldArena. EA-WM uses joint generation without ground-truth future trajectories as inference inputs.

	Physics Adherence		3D Accuracy		Controllability		P3CScore	**EWMScore**
Model	Interaction Quality	Trajectory Accuracy	Depth Accuracy	Perspectivity	Instruction Following	Semantic Alignment		
CogVideoX	0.594	0.353	0.910	0.783	0.727	**0.898**	71.08	57.91
WoW	0.556	0.206	0.728	0.767	0.569	0.884	61.83	54.87
Wan 2.2	0.518	0.163	0.777	0.766	0.538	0.888	60.83	54.55
TesserAct	0.580	0.140	0.716	0.792	0.615	0.878	62.02	53.23
GigaWorld-0	0.537	0.155	0.632	0.760	0.616	0.859	59.32	53.39
Vidar	0.535	0.193	0.787	0.759	0.591	0.883	62.47	51.61
Cosmos-Predict 2.5 (text)	0.387	0.082	0.705	0.796	0.266	0.773	50.15	50.80
Genie Envisioner	0.205	0.068	0.866	0.528	0.203	0.854	45.40	43.65
EA-WM (Ours)	**0.682**	**0.430**	**0.959**	**0.838**	**0.792**	0.895	**76.60**	**60.38**

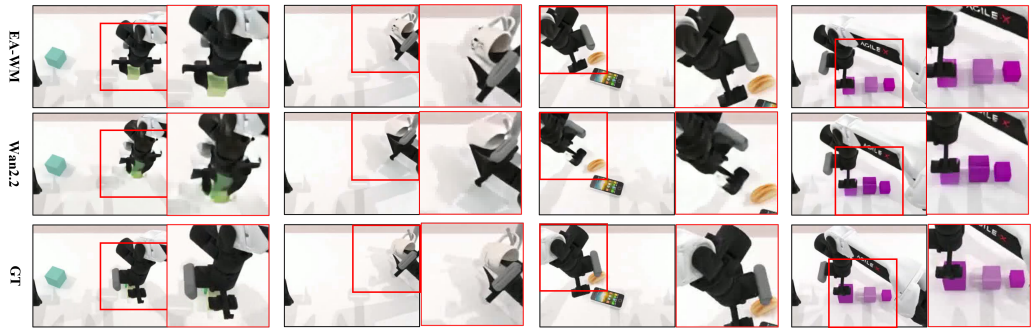

Figure 3: Qualitative comparison on four randomly selected RoboTwin 2.0 tasks. For each task, we visualize results at the same temporal step. Red boxes highlight local robot-object interaction regions.

Accuracy and Perspectivity by 0.049 and 0.042 over the strongest baselines, respectively, indicating stronger 3D consistency. Overall, EA-WM outperforms all compared models on five out of six selected metrics, demonstrating stronger physics adherence, 3D consistency, and controllability for robotic video world modeling.

Qualitative Analysis. We further conduct qualitative comparisons on four randomly selected tasks, visualizing GT, the Wan2.2 baseline, and EA-WM at the same temporal step for each task. As shown in Figure 3, we compare the results in four aspects: robot-object interaction, preservation of object geometry, physical consistency of robots, and completion of tasks. Compared with the Wan2.2 baseline, EA-WM jointly models KVAF and video streams through event-aware bidirectional fusion, allowing the model to better follow action-induced motion cues and focus on regions where object states and robot-object interactions change. As a result, EA-WM generates more accurate gripper-object contact relations, better preserves object identity and local geometry, and maintains more physically plausible robot poses during manipulation. Overall, EA-WM is closer to GT in both interaction fidelity and task progress, demonstrating stronger action-consistent future video generation.

4.3 Ablation Study

To verify the effectiveness of each component in EA-WM, we conduct ablation studies on the same selected WorldArena metrics used in quantitative evaluation. We compare the full model with three variants: the original Wan2.2 backbone, a variant that replaces KVAFs with numerical action representations, and a variant that removes event-aware fusion while keeping bidirectional cross-

Table 2: Ablation study on selected WorldArena metrics. All metrics are normalized to $[0, 1]$ and higher is better. P3CScore is the average over the six selected metrics multiplied by 100.

	Physics Adherence		3D Accuracy		Controllability		P3CScore
Variant	Interaction Quality	Trajectory Accuracy	Depth Accuracy	Perspectivity	Instruction Following	Semantic Alignment	
Wan2.2	0.518	0.163	0.777	0.766	0.538	0.888	60.83
w/o KVAFs	0.670	0.298	0.884	0.791	0.726	0.889	70.97
w/o EAF	0.656	0.407	0.947	0.800	0.782	**0.896**	74.80
EA-WM	**0.682**	**0.430**	**0.959**	**0.838**	**0.792**	0.895	**76.60**



Figure 4: Ablation visualization on the `ranking_block_size` task.

attention for video–KVAF interaction. We report Physics Adherence, 3D Accuracy, Controllability, and the averaged P3CScore.

The ablation results reveal the complementary roles of KVAFs and EAF. Compared with the variant without KVAFs, the full EA-WM improves P3CScore by 5.63 points, with particularly clear gains on Trajectory Accuracy and Depth Accuracy. This suggests that camera-aligned KVAFs provide useful geometric motion cues beyond numerical action inputs, especially for preserving robot trajectories and 3D structure in the generated videos. Compared with the variant without EAF, EA-WM improves P3CScore by 1.80 points and shows notable gains on Interaction Quality and Perspectivity, indicating that EDLS-guided event-aware fusion helps the model better focus on motion changes and robot-object interaction regions during cross-stream information exchange. Although the variant without EAF slightly outperforms EA-WM on Semantic Alignment, the full model achieves the best overall score and more balanced performance across physics adherence, 3D accuracy, and controllability.

We further visualize the `ranking_block_size` task to more intuitively compare the differences among ablation variants. As shown in Figure 4, the two ablations exhibit complementary failure modes. The variant without KVAFs can better maintain the local relations between the robot arm and objects, but it is less accurate in modeling the grasping trajectory and action-induced motion. In contrast, the variant without event-aware fusion can better infer the grasping route of the robot arm from KVAFs, but it struggles to preserve object consistency, such as the relative shape, size, and spatial arrangement of the blocks. By combining KVAFs with event-aware fusion, EA-WM handles both aspects more effectively: it follows the spatial motion pattern of the robot while preserving object consistency and robot-object interaction fidelity, generating videos that better match the physical progression of the task.

4.4 Additional Analysis

Action recovery and image-domain alignment of KVAFs. To better understand the information encoded in KVAFs, we conduct a preliminary action-recovery analysis in the numerical action space.

Table 3: Numerical action recovery from generated KVAFs. Errors are reported for translation, rotation, and gripper actions; lower is better.

Method	Translation Error	Rotation Error	Gripper Error
Raw-action baseline	**0.004**	**0.009**	**0.013**
KVAF recovery	0.0155	0.110	0.039

Heuristic KVAF recovery detection rate: ~ 0.45.

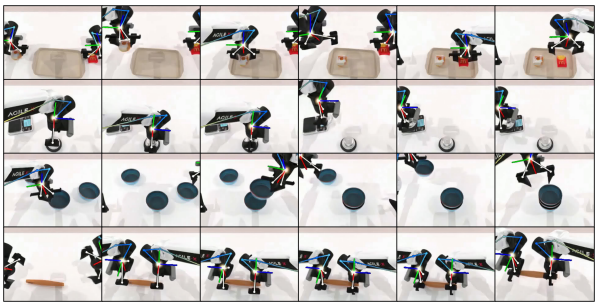

Figure 5: KVAFs overlays on generated videos. The overlays align with generated robot motion and reveal interpretable spatial cues learned from KVAFs.

Table 4: Comparison with action- or trajectory-conditioned baselines on selected WorldArena metrics. All metrics are normalized to $[0, 1]$ and higher is better. P3CScore is the average over the six selected metrics multiplied by 100.

	Physics Adherence		3D Accuracy		Controllability		P3CScore
Model	Interaction Quality	Trajectory Accuracy	Depth Accuracy	Perspectivity	Instruction Following	Semantic Alignment	
CtrlWorld	0.621	0.477	0.930	0.796	0.727	0.891	74.03
IRASim	0.566	0.364	0.931	0.779	0.660	0.885	69.75
Cosmos-Predict 2.5 (action)	0.550	0.295	0.886	0.764	0.584	0.888	66.12
RoboMaster	0.536	0.116	0.834	0.759	0.577	0.876	61.63
KVAF-conditioned generation	**0.682**	**0.494**	**0.977**	**0.850**	**0.784**	**0.901**	**78.13**

We use a heuristic procedure to recover end-effector translation, orientation, and gripper states from generated KVAF videos, and compare the recovered actions with predictions from a raw-action baseline that directly predicts numerical actions. As shown in Table 3, numerical action prediction still performs better on translation, rotation, and gripper errors. This gap is expected, since KVAFs are designed as image-domain action fields rather than direct numerical action outputs, and our current recovery pipeline depends on detecting small visual cues such as heatmap centers, pose axes, and gripper geometry. The detection rate of the heuristic KVAF recovery is approximately 0.45, suggesting that mapping structured visual action fields back to precise numerical actions remains a challenging problem. Nevertheless, the recovery results indicate that KVAFs contain partially recoverable action information, and motivate future work on more reliable bridges between image-domain action fields and numerical action spaces. More importantly, the overlay visualizations in Figure 5 show that the predicted KVAFs are well aligned with the generated robot motion across randomly selected tasks. The projected skeletons, end-effector cues, and pose axes follow the robot configuration in the generated videos, indicating that the network has learned the spatial and physical information contained in KVAFs rather than treating them as abstract conditioning signals. This image-domain alignment helps connect action-induced robot motion with object locations and local interaction regions, providing an interpretable bridge between low-dimensional actions and future video generation.

Reference-KVAF-conditioned video generation. We additionally evaluate EA-WM-Cond, a separately trained variant that receives an initial image, a text instruction, and reference KVAFs constructed from a robot trajectory. Unlike EA-WM-Joint, this variant uses KVAFs as external conditions rather than predicting their future sequence; the KVAF prediction head is removed. EA-WM-Cond achieves a P3CScore of 78.13 and an EWMScore of 60.84 (Tables 4 and 7). It performs best on the six selected robot-centric metrics among the compared action- or trajectory-conditioned baselines. Since these external baselines differ in architecture and conditioning interface, this is a benchmark-level comparison, not a representation-only controlled experiment.

4.5 Computational Cost and Real-World Evaluation

Computational cost. EA-WM contains 10.40B DiT-side parameters, compared with 5.00B for Wan2.2-TI2V-5B; these counts exclude the shared VAE and text encoder. Under the matched A100 setup in Sec. A.9, RGB-generation latency increases from 49.30 s to 96.38 s (+95.5%), and peak allocated memory increases from 24.79 GiB to 34.88 GiB (+40.7%). The additional cost mainly comes from the full-depth KVAF stream.

Offline real-world evaluation. On RH20T [6], using 2,000 post-training clips and 10 held-out episodes from a UR5 robot with a WSG-50 gripper and one fixed camera, EA-WM achieves an EWMScore of 57.59, compared with 56.51 for post-trained Wan2.2. This is a preliminary offline real-world evaluation, not closed-loop deployment on a physical robot. Full results are provided in Sec. A.11.

5 Limitations

EA-WM requires robot kinematics, camera calibration, and synchronized action-state logs to construct KVAFs. While this yields accurate, well-aligned action representations in simulation, real-world deployment may be affected by calibration errors, occlusions, sensor noise, or embodiment shifts. Current KVAFs mainly encode robot-side geometric motion cues, while object-state changes and robot-object interactions are captured indirectly through event-aware fusion and EDLS. Extending KVAFs with object-centric or contact-aware visual fields is a promising future direction.

6 Conclusion

We present **EA-WM**, an event-aware generative world model for action-consistent robot-video generation. Unlike current WAMs that mainly optimize action prediction or policy learning, EA-WM focuses on preserving robot spatial motion and robot-object interaction dynamics in generated videos. It maps actions and kinematic states into *Structured Kinematic-to-Visual Action Fields* (KVAFs) aligned with the visual generation domain, and integrates them through EDLS-guided event-aware bidirectional fusion. On WorldArena, EA-WM achieves the best $\mathrm{P3CScore}$ among compared models.

Acknowledgments and Disclosure of Funding

This work was conducted during an internship at DeepCybo. We thank DeepCybo for providing computational resources and research support. This work received no specific external research grant.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[2] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, Yuanzhen Li, Michael Rubinstein, Tomer Michaeli, Oliver Wang, Deqing Sun, Tali Dekel, and Inbar Mosseri. Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*, 2024.

[3] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[5] Yixiang Chen, Peiyan Li, Jiabing Yang, Keji He, Xiangnan Wu, Yuan Xu, Kai Wang, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. BridgeV2W: Bridging video generation models to embodied world models via embodiment masks. *arXiv preprint arXiv:2602.03793*, 2026.

[6] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[7] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.

[8] Xin Gao, Li Hu, Siqi Hu, Mingyang Huang, Chaonan Ji, Dechao Meng, Jinwei Qi, Penchong Qiao, Zhen Shen, Yafei Song, Ke Sun, Linrui Tian, Guangyuan Wang, Qi Wang, Zhongjian Wang, Jiayu Xiao, Sheng Xu, Bang Zhang, Peng Zhang, Xindi Zhang, Zhe Zhang, Jingren Zhou, and Lian Zhuo. Wan-S2V: Audio-driven cinematic video generation. *arXiv preprint arXiv:2508.18621*, 2025.

[9] 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.

[10] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[11] Peiyan Li, Yixiang Chen, Yuan Xu, Jiabing Yang, Xiangnan Wu, Jun Guo, Nan Sun, Long Qian, Xinghang Li, Xin Xiao, Jing Liu, Nianfeng Liu, Tao Kong, Yan Huang, Liang Wang, and Tieniu Tan. Multi-view video diffusion policy: A 3D spatio-temporal-aware video action model. *arXiv preprint arXiv:2604.03181*, 2026.

[12] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[13] Yaxuan Li, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Worldeval: World model as real-world robot policies evaluator. *arXiv preprint arXiv:2505.19017*, 2025.

[14] 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.

[15] OpenAI. Video generation models as world simulators. `https://openai.com/index/video-generation-models-as-world-simulators/`, 2024. Technical report/blog post.

[16] 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.

[17] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[18] Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation. *arXiv preprint arXiv:2506.00613*, 2025.

[19] Marc Rigter, Tarun Gupta, Agrin Hilmkil, and Chao Ma. AVID: Adapting video diffusion models to world models. *arXiv preprint arXiv:2410.12822*, 2024.

[20] Sandeep Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions. *arXiv preprint arXiv:2511.07732*, 2025.

[21] Yu Shang, Zhuohang Li, Yiding Ma, Weikang Su, Xin Jin, Ziyou Wang, Lei Jin, Xin Zhang, Yinzhou Tang, Haisheng Su, Gao Chen, Wei Wu, Xihui Liu, Dhruv Shah, Zhaoxiang Zhang, Zhibo Chen, Jun Zhu, Yonghong Tian, Tat-Seng Chua, Wenwu Zhu, and Yong Li. WorldArena: A unified benchmark for evaluating perception and functional utility of embodied world models. *arXiv preprint arXiv:2602.08971*, 2026.

[22] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025.

[23] Mohit Shridhar, Yat Long Lo, and Stephen James. Generative image as action models. *arXiv preprint arXiv:2407.07875*, 2024.

[24] Yue Su, Sijin Chen, Haixin Shi, Mingyu Liu, Zhengshen Zhang, Ningyuan Huang, Weiheng Zhong, Zhengbang Zhu, Yuxiao Liu, and Xihui Liu. World guidance: World modeling in condition space for action generation. *arXiv preprint arXiv:2602.22010*, 2026.

[25] Wei-Cheng Tseng, Jinwei Gu, Qinsheng Zhang, Hanzi Mao, Ming-Yu Liu, Florian Shkurti, and Yen-Chen Lin. Scalable policy evaluation with video world models. *arXiv preprint arXiv:2511.11520*, 2025.

[26] Vitalis Vosylius, Younggyo Seo, Jafar Uruç, and Stephen James. Render and diffuse: Aligning image and action spaces for diffusion-based behaviour cloning. *arXiv preprint arXiv:2405.18196*, 2024.

[27] Wan Team. Wan2.2: Open and advanced large-scale video generative models. `https://github.com/Wan-Video/Wan2.2`, 2025. GitHub repository and model release.

[28] Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[29] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive videogpts are scalable world models. In *Advances in Neural Information Processing Systems*, 2024.

[30] Mutian Xu, Tianbao Zhang, Tianqi Liu, Zhaoxi Chen, Xiaoguang Han, and Ziwei Liu. Kinema4D: Kinematic 4D world modeling for spatiotemporal embodied simulation. *arXiv preprint arXiv:2603.16669*, 2026.

[31] Yuan Xu, Jiabing Yang, Xiaofeng Wang, Yixiang Chen, Zheng Zhu, Bowen Fang, Guan Huang, Xinze Chen, Yun Ye, Qiang Zhang, Peiyan Li, Xiangnan Wu, Kai Wang, Bing Zhan, Shuo Lu, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. EgoDemoGen: Egocentric demonstration generation for viewpoint generalization in robotic manipulation. *arXiv preprint arXiv:2509.22578*, 2025.

[32] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, 2025.

[33] 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.

[34] Haoyu Zhen, Zixian Gao, Qiao Sun, Yilin Zhao, Yuncong Yang, Yilun Du, Pengsheng Guo, Tsun-Hsuan Wang, Yi-Ling Qiao, and Chuang Gan. Action images: End-to-end policy learning via multiview video generation. *arXiv preprint arXiv:2604.06168*, 2026.

[35] Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. *arXiv preprint arXiv:2501.10105*, 2025.

[36] 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.

[37] Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

A Appendix

A.1 Detailed Network Architecture Pipeline

This section describes the full EA-WM network pipeline. We focus on how RGB videos, KVAFs, and event-difference targets are processed by the Wan2.2-based architecture.

Latent preparation. Given an RGB video sequence $\mathbf{X} = \{I_\tau\}_{\tau=1}^{T}$, the corresponding KVAF sequence $\mathbf{K}$, and text condition $\mathbf{c}$, EA-WM first encodes both videos and KVAFs with the same pretrained video VAE:

$$\mathbf{z}_0^v = \mathrm{VAE}(\mathbf{X}), \qquad \mathbf{z}_0^k = \mathrm{VAE}(\mathbf{K}).$$

This places the target video and action-derived KVAFs in a shared latent space. During flow-matching training, both latents are perturbed at timestep t, producing noised latents $\mathbf{z}_t^v$ and $\mathbf{z}_t^k$ together with their training targets $\mathbf{Y}^v$ and $\mathbf{Y}^k$.

Dual-stream denoising. After patchification, the video latent and KVAF latent are represented as two token streams,

$$\mathbf{H}_0^v = \mathcal{P}(\mathbf{z}_t^v), \qquad \mathbf{H}_0^k = \mathcal{P}(\mathbf{z}_t^k).$$

The video stream follows the original Wan2.2 DiT blocks, while the KVAF stream is processed by a full-depth copy of the DiT blocks. Event-aware fusion modules are inserted at a sparse set of layers $\mathcal{S}$, allowing the two streams to exchange information only at selected depths. This keeps the KVAF branch expressive while avoiding dense, overly expensive cross-stream interaction.

Event-Difference Latent Supervision. EA-WM uses frame-difference latents as event supervision. Given the input video, we compute

$$\Delta I_1 = \mathbf{0}, \qquad \Delta I_\tau = |I_\tau - I_{\tau-1}|, \quad \tau = 2, \dots, T,$$

and encode the difference sequence with the same video VAE:

$$\mathbf{E} = \mathrm{VAE}(\{\Delta I_\tau\}_{\tau=1}^{T}).$$

This target supervises the event prediction branch in each fusion module. Since the event prediction and event gate share the same event representation, EDLS also shapes the gate that controls video–KVAF cross-attention.

Algorithm 1 EA-WM training step

Require: RGB video $\mathbf{X}$, KVAF sequence $\mathbf{K}$, text condition $\mathbf{c}$, fusion layers $\mathcal{S}$

Encode latents: $\mathbf{z}_0^v \leftarrow \mathrm{VAE}(\mathbf{X})$, $\mathbf{z}_0^k \leftarrow \mathrm{VAE}(\mathbf{K})$

Sample timestep t and construct noised latents and flow targets:

$$(\mathbf{z}_t^v, \mathbf{Y}^v),\ (\mathbf{z}_t^k, \mathbf{Y}^k) \leftarrow \mathrm{FlowMatch}(\mathbf{z}_0^v, \mathbf{z}_0^k, t)$$

Set $\Delta I_1 \leftarrow \mathbf{0}$ and $\Delta I_\tau \leftarrow |I_\tau - I_{\tau-1}|$ for $\tau = 2, \dots, T$

Encode the EDLS target: $\mathbf{E} \leftarrow \mathrm{VAE}(\{\Delta I_\tau\}_{\tau=1}^{T})$

Patchify: $\mathbf{H}_0^v \leftarrow \mathcal{P}(\mathbf{z}_t^v)$, $\mathbf{H}_0^k \leftarrow \mathcal{P}(\mathbf{z}_t^k)$

for $\ell = 1$ to L **do**

  Update KVAF stream: $\bar{\mathbf{H}}_\ell^k \leftarrow \mathcal{B}_\ell(\mathbf{H}_{\ell-1}^k, \mathbf{c}, t)$

  if $\ell \in \mathcal{S}$ **then**

    $\tilde{\mathbf{H}}_{\ell-1}^v, \mathbf{H}_\ell^k, \hat{\mathbf{E}}_\ell \leftarrow \mathrm{EventFusion}_\ell(\mathbf{H}_{\ell-1}^v, \bar{\mathbf{H}}_\ell^k)$

  else

    $\tilde{\mathbf{H}}_{\ell-1}^v \leftarrow \mathbf{H}_{\ell-1}^v, \quad \mathbf{H}_\ell^k \leftarrow \bar{\mathbf{H}}_\ell^k$

  end if

  Update video stream: $\mathbf{H}_\ell^v \leftarrow \mathcal{D}_\ell(\tilde{\mathbf{H}}_{\ell-1}^v, \mathbf{c}, t)$

end for

Predict $\hat{\mathbf{Y}}^v \leftarrow \mathrm{Head}_v(\mathbf{H}_L^v), \quad \hat{\mathbf{Y}}^k \leftarrow \mathrm{Head}_k(\mathbf{H}_L^k)$

Compute $\mathcal{L} = \omega(t)(\|\hat{\mathbf{Y}}^v - \mathbf{Y}^v\|_2^2 + \|\hat{\mathbf{Y}}^k - \mathbf{Y}^k\|_2^2) + \lambda_{\mathrm{evt}}\mathcal{L}_{\mathrm{EDLS}}$

return $\mathcal{L}$

Event-aware bidirectional fusion. The fusion block is the key interaction module between the video stream and the KVAF stream. It first predicts a shared event representation, from which both the event gate and the event latent are derived. The event latent is supervised by EDLS, while the event gate modulates the strength of bidirectional cross-attention. Therefore, EDLS affects video generation not only through an auxiliary prediction loss, but also through the gate that controls information exchange between the two streams.

Algorithm 2 Event-aware bidirectional fusion at layer ℓ

Require: Video tokens $\mathbf{H}^v_{\ell-1}$, KVAF tokens $\bar{\mathbf{H}}^k_\ell$

1: Compute shared event features:

$$\mathbf{M}_\ell \leftarrow \Phi_\ell(\mathbf{H}^v_{\ell-1}, \bar{\mathbf{H}}^k_\ell)$$

2: Predict the gate and event latent:

$$\mathbf{G}_\ell \leftarrow \Gamma_\ell(\mathbf{M}_\ell), \qquad \hat{\mathbf{E}}_\ell \leftarrow \Psi_\ell(\mathbf{M}_\ell)$$

3: Video reads KVAF:

$$\mathbf{R}^v_\ell \leftarrow \mathrm{CA}_{v\leftarrow k}(\mathbf{H}^v_{\ell-1}, \bar{\mathbf{H}}^k_\ell)$$

4: KVAF reads video:

$$\mathbf{R}^k_\ell \leftarrow \mathrm{CA}_{k\leftarrow v}(\bar{\mathbf{H}}^k_\ell, \mathbf{H}^v_{\ell-1})$$

5: Apply event-gated residual fusion:

$$\tilde{\mathbf{H}}^v_{\ell-1} = \mathbf{H}^v_{\ell-1} + \mathbf{G}_\ell \odot \mathbf{R}^v_\ell$$

$$\mathbf{H}^k_\ell = \bar{\mathbf{H}}^k_\ell + \mathbf{G}_\ell \odot \mathbf{R}^k_\ell$$

6: **return** $\tilde{\mathbf{H}}^v_{\ell-1}, \mathbf{H}^k_\ell, \hat{\mathbf{E}}_\ell$

Training objective. The full training loss consists of video flow matching, KVAF flow matching, and EDLS:

$$\mathcal{L} = \omega(t)\left(\left\|\hat{\mathbf{Y}}^v - \mathbf{Y}^v\right\|_2^2 + \left\|\hat{\mathbf{Y}}^k - \mathbf{Y}^k\right\|_2^2\right) + \lambda_{\mathrm{evt}} \frac{1}{|\mathcal{S}|} \sum_{\ell\in\mathcal{S}} \left\|\mathrm{Unpatchify}(\hat{\mathbf{E}}_\ell) - \mathbf{E}\right\|_2^2.$$

The video loss trains the future video generation branch, the KVAF loss encourages the KVAF branch to preserve structured robot geometry, and EDLS guides the fusion modules to focus on motion and interaction changes.

A.2 KVAF Construction

This section describes how we construct *Structured Kinematic-to-Visual Action Fields* (KVAFs) from robot actions and kinematic states. KVAFs are designed to convert low-dimensional robot-side information into camera-aligned visual action fields, so that action conditioning can be expressed in the same image domain as future video generation.

Inputs. For each episode, we assume access to the robot URDF, embodiment configuration, camera parameters, and per-frame robot states. At time step t, the input to KVAF construction is

$$\mathcal{I}_t = \left(\mathbf{q}^L_t, \mathbf{q}^R_t, g^L_t, g^R_t, \boldsymbol{\xi}^L_t, \boldsymbol{\xi}^R_t, \mathbf{K}_t, \mathbf{E}_t\right),$$

where $\mathbf{q}^L_t, \mathbf{q}^R_t$ are the left/right arm joint values, g^L_t, g^R_t are normalized gripper states, $\boldsymbol{\xi}^L_t, \boldsymbol{\xi}^R_t$ are end-effector poses, and $(\mathbf{K}_t, \mathbf{E}_t)$ are camera intrinsics and extrinsics. Each end-effector pose is represented by a 3D position and a quaternion:

$$\boldsymbol{\xi}^a_t = (\mathbf{p}^a_t, \mathbf{q}^a_t), \qquad a \in \{L, R\}.$$

Kinematic lifting. We first lift low-dimensional joint states into 3D robot geometry using forward kinematics. Let $\mathbf{T}^W_{a,k}(t) \in SE(3)$ denote the world-frame transform of the k-th link of arm a at time t. Starting from the robot base transform, each link transform is computed recursively:

$$\mathbf{T}^W_{a,k}(t) = \mathbf{T}^W_{a,k-1}(t)\mathbf{T}^{\mathrm{orig}}_{a,k}\mathbf{T}^{\mathrm{mot}}_{a,k}(q^a_{t,k}),$$

where $\mathbf{T}_{a,k}^{\text{orig}}$ is the static joint transform parsed from the URDF, and $\mathbf{T}_{a,k}^{\text{mot}}$ is the revolute or prismatic motion induced by the joint value. For the gripper, the normalized signal is mapped to a prismatic displacement

$$d_t^a = \text{clip}(g_t^a, 0, 1) d_{\max}^a,$$

and used to compute the two finger branches. This gives a set of 3D robot keypoints

$$\mathcal{P}_t^a = \{\mathbf{p}_{a,k}^W(t)\}_{k=1}^{K_a},$$

including the arm skeleton, joint landmarks, and gripper endpoints.

Camera projection. Each 3D keypoint is projected to the target camera view. For a world-frame point $\mathbf{p}^W$, we compute

$$\mathbf{p}^C = \mathbf{E}_t \begin{bmatrix} \mathbf{p}^W \\ 1 \end{bmatrix}, \qquad \hat{\mathbf{u}} = \mathbf{K}_t \mathbf{p}_{1:3}^C,$$

and obtain the image coordinate by perspective division:

$$(u, v) = \left(\frac{\hat{u}_x}{\hat{u}_z}, \frac{\hat{u}_y}{\hat{u}_z} \right).$$

Points with non-positive camera depth are discarded. This step maps robot kinematics from the world frame to the exact image plane used by the video world model.

Visual rendering. Given projected keypoints, we render a KVAF frame $\mathbf{V}_t \in \mathbb{R}^{H \times W \times 3}$ on a black canvas. The rendered elements include depth-aware arm skeletons, joint landmarks, gripper geometry, end-effector heatmaps, and pose axes. For depth-aware skeleton rendering, we encode the absolute camera-space depth z, measured in meters along the camera's optical axis, using fixed bounds $z_{\text{near}} = 0.05\,\text{m}$ and $z_{\text{far}} = 0.70\,\text{m}$. These bounds are shared across frames and episodes in our RoboTwin KVAF construction:

$$\alpha(z) = \text{clip}\left(\frac{z - z_{\text{near}}}{z_{\text{far}} - z_{\text{near}}}, 0, 1 \right), \qquad i(z) = \text{round}\left(255[1 - \alpha(z)]\right), \qquad c(z) = \mathcal{P}_{i(z)},$$

where $\mathcal{P}$ is a fixed 256-entry depth color palette. This mapping provides absolute camera-depth cues with a consistent metric scale across episodes. Depths outside the specified range are clipped to the corresponding endpoint colors.

For the end-effector, we render a truncated Gaussian heatmap centered at the projected end-effector location (u_t^a, v_t^a):

$$H_t^a(x, y) = \exp\left(-\frac{(x - u_t^a)^2 + (y - v_t^a)^2}{2\sigma^2} \right) \mathbf{1}\left[(x - u_t^a)^2 + (y - v_t^a)^2 \le r^2 \right].$$

This heatmap provides an explicit cue for the manipulation focus. We further render the local end-effector pose axes. Given the rotation matrix $\mathbf{R}_t^a$ from the end-effector quaternion, each local axis endpoint is

$$\mathbf{p}_t^{a,m} = \mathbf{p}_t^a + \ell_{\text{axis}} \mathbf{R}_t^a \mathbf{e}_m, \qquad m \in \{x, y, z\}.$$

After projection, the three axes are drawn in RGB colors. The final KVAF frame is the composition of all rendered components:

$$\mathbf{V}_t = \text{Render}\left(\mathcal{P}_t^L, \mathcal{P}_t^R, \boldsymbol{\xi}_t^L, \boldsymbol{\xi}_t^R, \mathbf{K}_t, \mathbf{E}_t \right).$$

Stacking all frames gives the KVAF sequence

$$\mathcal{V} = \{\mathbf{V}_t\}_{t=1}^T.$$

In the current implementation, KVAFs explicitly encode robot-side geometric motion information. Object-state changes are not directly rasterized in KVAFs, but are later captured through event-aware fusion and EDLS in the generative model.

Algorithm 3 KVAF construction for one episode

Require: Robot URDF, embodiment config, episode states $\{\mathcal{I}_t\}_{t=1}^{T}$, image size (H, W)
Ensure: KVAF sequence $\mathcal{V} = \{\mathbf{V}_t\}_{t=1}^{T}$

Parse URDF to obtain joint origins, joint axes, link hierarchy, and gripper joints
Build left/right arm kinematic chains from the robot configuration
Set fixed camera-depth bounds $z_{\text{near}} \leftarrow 0.05\,\text{m}$ and $z_{\text{far}} \leftarrow 0.70\,\text{m}$
for $t = 1$ to T **do**
    Initialize a black canvas $\mathbf{V}_t \in \mathbb{R}^{H \times W \times 3}$
    for each arm $a \in \{L, R\}$ **do**
        Compute 3D arm and gripper keypoints by forward kinematics:

$$\mathcal{P}_t^a \leftarrow \text{FK}(\mathbf{q}_t^a, g_t^a, \text{URDF})$$

        Project keypoints to the target camera:

$$\Pi_t^a \leftarrow \text{Project}(\mathcal{P}_t^a, \mathbf{K}_t, \mathbf{E}_t)$$

        Render depth-aware arm skeleton and joint landmarks on $\mathbf{V}_t$
        Render gripper geometry using projected finger keypoints
        Project end-effector center and render Gaussian heatmap
        Project end-effector pose axes and render RGB orientation cues
    end for
    Append $\mathbf{V}_t$ to $\mathcal{V}$
end for
return $\mathcal{V}$

Table 5: Rendered components in KVAFs and their roles.

Component	Encoded information
Depth-aware skeleton	Arm topology, spatial position, and absolute camera-depth cues
Joint landmarks	Local joint locations and articulated structure
Gripper geometry	End-effector opening state and finger configuration
End-effector heatmap	Manipulation focus and target interaction region
Pose axes	End-effector orientation and approach direction

A.3 Heuristic Action Recovery from KVAFs

This section describes the heuristic procedure used in our additional analysis to recover numerical actions from generated KVAF videos. This procedure is not part of EA-WM training or inference; it is only used to examine whether the visual action fields contain recoverable low-dimensional action information. Given a generated KVAF video, camera intrinsics and extrinsics, and the known rendered pose-axis length, we recover end-effector trajectories, gripper states, and then convert them into RoboTwin-style relative actions.

End-effector detection and pose recovery. For each KVAF frame, we first detect candidate end-effector centers from high-intensity heatmap regions. Around each detected center, we identify the rendered RGB pose-axis endpoints by thresholding local red, green, and blue regions. Let the detected image points be

$$\mathbf{u}_0, \mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z \in \mathbb{R}^2,$$

where $\mathbf{u}_0$ is the end-effector center and $\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z$ correspond to the projected local x, y, z axes. We associate them with the canonical 3D local points

$$\mathbf{o}_0 = \mathbf{0}, \qquad \mathbf{o}_x = \ell_{\text{axis}}\mathbf{e}_x, \qquad \mathbf{o}_y = \ell_{\text{axis}}\mathbf{e}_y, \qquad \mathbf{o}_z = \ell_{\text{axis}}\mathbf{e}_z.$$

Using the camera intrinsic matrix, we solve a PnP problem to estimate the end-effector pose in the camera frame, and then transform it to the world frame using the camera extrinsics. This gives a recovered pose

$$\hat{\boldsymbol{\xi}}_t^a = (\hat{\mathbf{p}}_t^a, \hat{\mathbf{R}}_t^a)$$

for each detected arm a at time t.

Gripper state and bimanual track association. The gripper state is estimated from the local gripper geometry around the recovered end-effector center. We detect gripper-like white components near the pose axes and use their relative separation as a raw gripper opening signal. Since RoboTwin contains bimanual manipulation, detections are associated into left and right tracks. When two detections are present, we assign them by their recovered world-frame horizontal position. When only one detection is available, we assign it to the closer existing track. Missing detections are filled by nearest-neighbor interpolation, and the recovered pose and gripper sequences are smoothed with a short temporal median filter. Finally, the gripper curve is normalized to $[0, 1]$ within each episode.

Conversion to relative actions. After obtaining dense left and right end-effector pose sequences, we convert them into RoboTwin-style relative actions. For each arm $a \in \{L, R\}$, the local translation action is computed in the current end-effector frame:

$$\Delta \mathbf{p}_t^a = (\hat{\mathbf{R}}_t^a)^\top \left(\hat{\mathbf{p}}_{t+1}^a - \hat{\mathbf{p}}_t^a\right).$$

The relative rotation is

$$\Delta \mathbf{R}_t^a = (\hat{\mathbf{R}}_t^a)^\top \hat{\mathbf{R}}_{t+1}^a,$$

which is converted to roll-pitch-yaw angles:

$$\Delta \boldsymbol{\theta}_t^a = \mathrm{RPY}(\Delta \mathbf{R}_t^a).$$

The gripper action is the temporal difference of the normalized gripper state:

$$\Delta g_t^a = \hat{g}_{t+1}^a - \hat{g}_t^a.$$

Thus the recovered bimanual action at time t is

$$\hat{\mathbf{a}}_t = [\Delta \mathbf{p}_t^L, \Delta \boldsymbol{\theta}_t^L, \Delta g_t^L, \Delta \mathbf{p}_t^R, \Delta \boldsymbol{\theta}_t^R, \Delta g_t^R].$$

Evaluation. We compare the recovered actions with ground-truth action CSVs using group-wise errors for translation, rotation, and gripper commands. Translation and rotation errors are computed as vector norms over the corresponding action dimensions, while the gripper error is computed as absolute error. Because the recovery depends on detecting small rendered cues such as heatmap centers, pose axes, and gripper components, the current heuristic pipeline is not expected to match direct numerical action prediction. Instead, it serves as an analysis tool for probing how much action information can be recovered from the image-domain KVAF representation.

The relatively low detection rate of this heuristic recovery pipeline reflects the difficulty of inverting image-domain action fields back into precise numerical actions. This supports our view that KVAFs are better suited as a video-domain conditioning interface rather than as a direct replacement for numerical action labels.

A.4 Full-metric evaluation

In the main paper, we focus on Physics Adherence, 3D Accuracy, and Controllability, as these dimensions are most directly related to robot motion, spatial consistency, and action-conditioned generation. For completeness, we additionally report all 16 WorldArena video perception metrics in Tables 6 and 7. Table 6 compares EA-WM with the representative video and embodied world models used in the main quantitative analysis, while Table 7 compares KVAF-conditioned generation with mainstream action- or trajectory-conditioned baselines. Unlike the main-paper $\mathrm{P3CScore}$, which is computed only on the six selected metrics, the appendix $\mathrm{EWMScore}$ averages all 16 metrics including Action Following. Across both comparisons, EA-WM and KVAF-conditioned generation achieve the strongest overall $\mathrm{EWMScore}$, while also showing consistent advantages on robot-centric metrics such as Interaction Quality, Trajectory Accuracy, Depth Accuracy, and Perspectivity. These results further support that representing actions as camera-aligned KVAFs provides effective structured guidance for robotic video world modeling.

A.5 Additional Visualizations of Generated KVAFs and Videos

To further examine the learned video-domain action representation, we visualize generated KVAF sequences together with the corresponding generated video frames in Figures 6–10. Across different tasks, the generated KVAFs are well aligned with the robot motion in the videos: skeletons, end-effector cues, gripper geometry, and pose axes follow the generated robot configurations, while the video frames maintain consistent object locations and interaction progress.

Algorithm 4 Heuristic action recovery from KVAF videos

Require: Generated KVAF video $\mathcal{V}$, camera intrinsics $\mathbf{K}$, camera extrinsics $\mathbf{E}$, pose-axis length ℓ_{axis}

Ensure: Recovered relative action sequence $\{\hat{\mathbf{a}}_t\}$

for each frame $\mathbf{V}_t$ **do**
  Detect candidate end-effector centers from heatmap peaks
  for each detected center **do**
    Detect local RGB pose-axis endpoints $(\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)$
    Solve PnP using canonical 3D axis points and detected 2D endpoints
    Convert the recovered camera-frame pose to world frame
    Estimate raw gripper opening from local gripper geometry
  end for
  Store all valid detections for frame t
end for
Associate detections into left/right tracks using recovered world positions
Fill missing detections by nearest-neighbor interpolation
Smooth pose and gripper trajectories with a temporal median filter
Normalize gripper curves to $[0, 1]$
for each arm $a \in \{L, R\}$ and timestep t **do**
  Compute local translation:

$$\Delta \mathbf{p}_t^a = (\hat{\mathbf{R}}_t^a)^\top (\hat{\mathbf{p}}_{t+1}^a - \hat{\mathbf{p}}_t^a)$$

  Compute local rotation:

$$\Delta \boldsymbol{\theta}_t^a = \mathrm{RPY}\left((\hat{\mathbf{R}}_t^a)^\top \hat{\mathbf{R}}_{t+1}^a\right)$$

  Compute gripper delta:

$$\Delta g_t^a = \hat{g}_{t+1}^a - \hat{g}_t^a$$

end for
Concatenate left/right translation, rotation, and gripper deltas:

$$\hat{\mathbf{a}}_t = [\Delta \mathbf{p}_t^L, \Delta \boldsymbol{\theta}_t^L, \Delta g_t^L, \Delta \mathbf{p}_t^R, \Delta \boldsymbol{\theta}_t^R, \Delta g_t^R]$$

return $\{\hat{\mathbf{a}}_t\}$

Table 6: Full WorldArena video perception metrics for the models compared in the main results. All metrics are normalized to $[0, 1]$ and higher is better. EWMScore averages all 16 metrics, and multiplies the result by 100.

	Visual Quality			Motion Quality			Content Consistency			Physics		3D		Controllability			EWMScore
Model	Img.	Aes.	JEPA	Dyn.	Flow	Smooth	Subj.	Bg.	Photo	Inter.	Traj.	Depth	Persp.	Instr.	Sem.	Action	
CogVideoX	0.358	0.378	0.938	0.317	0.219	0.739	0.808	0.877	0.358	0.594	0.353	0.910	0.783	0.727	**0.898**	0.008	57.91
WoW	0.459	0.387	0.744	0.461	0.271	0.769	0.816	0.903	0.217	0.556	0.206	0.728	0.767	0.569	0.884	0.043	54.87
Wan 2.2	0.388	0.396	0.758	0.435	0.127	0.702	**0.839**	0.904	**0.478**	0.518	0.163	0.777	0.766	0.538	0.888	0.051	54.55
TesserAct	0.332	**0.459**	0.458	0.515	0.245	0.758	0.825	**0.924**	0.249	0.580	0.140	0.716	0.792	0.615	0.878	0.031	53.23
GigaWorld-0	0.504	0.399	0.441	0.671	0.312	0.781	0.730	0.856	0.176	0.537	0.155	0.632	0.760	0.616	0.859	0.113	53.39
Vidar	0.415	0.407	0.561	0.277	0.143	**0.797**	0.763	0.830	0.235	0.535	0.193	0.787	0.759	0.591	0.883	0.082	51.61
Cosmos-Predict 2.5 (text)	**0.667**	0.450	0.313	0.591	**0.430**	0.788	0.749	0.851	0.138	0.387	0.082	0.705	0.796	0.266	0.773	**0.142**	50.80
Genie Envisioner	0.231	0.329	0.334	**0.693**	0.086	0.697	0.776	0.902	0.201	0.205	0.068	0.866	0.528	0.203	0.854	0.011	43.65
EA-WM (Ours)	0.364	0.387	**0.951**	0.360	0.273	0.730	0.827	0.904	0.220	**0.682**	**0.430**	**0.959**	**0.838**	**0.792**	0.895	0.048	**60.38**

Table 7: Full WorldArena video perception metrics for mainstream action- or trajectory-conditioned baselines. All metrics are normalized to $[0, 1]$ and higher is better. EWMScore averages all 16 metrics, and multiplies the result by 100.

	Visual Quality			Motion Quality			Content Consistency			Physics		3D		Controllability			EWMScore
Model	Img.	Aes.	JEPA	Dyn.	Flow	Smooth	Subj.	Bg.	Photo	Inter.	Traj.	Depth	Persp.	Instr.	Sem.	Action	
CtrlWorld	0.352	**0.389**	0.919	0.426	**0.345**	**0.738**	**0.841**	0.906	0.173	0.621	0.477	0.930	0.796	0.727	0.891	0.021	59.70
IRASim	0.349	0.362	0.933	0.414	0.208	0.705	0.831	0.907	0.352	0.566	0.364	0.931	0.779	0.660	0.885	**0.053**	58.12
Cosmos-Predict 2.5 (action)	**0.449**	0.358	0.930	0.399	0.057	0.710	0.820	0.889	**0.353**	0.550	0.295	0.886	0.764	0.584	0.888	0.013	55.91
RoboMaster	0.349	0.384	0.297	**0.612**	0.148	0.694	0.830	**0.912**	0.336	0.536	0.116	0.834	0.759	0.577	0.876	0.035	51.84
KVAF-conditioned generation	0.364	0.383	**0.950**	0.360	0.268	0.729	0.818	0.899	0.229	**0.682**	**0.494**	**0.977**	**0.850**	**0.784**	**0.901**	0.048	**60.84**



Figure 6: Generated KVAF and video frame sequence



Figure 7: Generated KVAF and video frame sequence



Figure 8: Generated KVAF and video frame sequence



Figure 9: Generated KVAF and video frame sequence

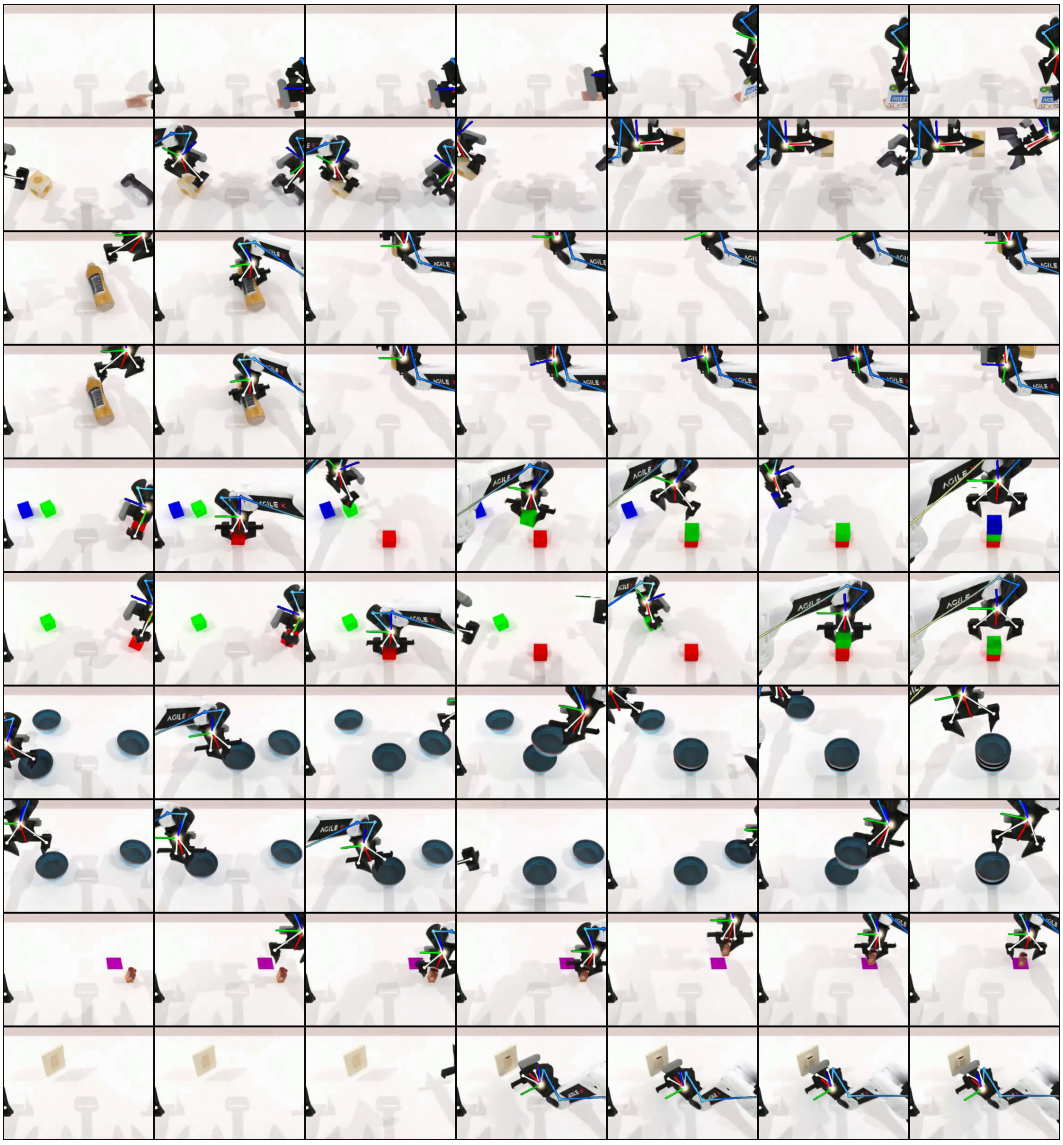

Figure 10: Generated KVAF and video frame sequence

A.6 Broader Impacts

EA-WM is primarily evaluated in simulation, with a preliminary offline evaluation on real-world RH20T data. The model is not deployed for closed-loop control on physical robots in this work. The model does not directly execute robot actions, nor is it deployed on real robots in this work. Nevertheless, more accurate robotic world models may eventually be used as predictive simulators, planning modules, policy-learning aids, or evaluation environments for downstream VLA systems. If such models generate misleading future rollouts, downstream policies or planners may make unsafe decisions, especially in physical environments involving humans, fragile objects, or safety-critical manipulation.

This work does not involve human-subject data, personally identifiable information, or real-robot deployment. The primary goal is to study how structured action information can improve robotic video world modeling.

A.7 Controlled Analysis of KVAF Representations

We further conduct controlled experiments to disentangle the effect of the KVAF representation from additional geometric information and model capacity. *Geometry49* contains the same robot kinematic states, end-effector poses, gripper states, and camera parameters used to construct KVAFs, but encodes them as a 49-dimensional numerical representation. We additionally consider two simplified image-space representations: *EEF Pose Map*, which retains only projected end-effector locations and orientations, and *Binary Embodiment Mask*, which rasterizes the projected robot structure as a binary mask. The visual representation variants use the same dual-stream architecture, fusion design, training configuration, and parameter budget, differing only in the information retained by the visual action representation.

Table 8: Full WorldArena comparison of KVAF with an information-matched numerical representation and simplified visual representations. All individual metrics are normalized to $[0, 1]$ and higher is better. EWMScore averages all 16 WorldArena metrics and is multiplied by 100; P3CScore averages the six robot-centric metrics used in the main paper and is also multiplied by 100. Aggregate scores are computed from the unrounded metric values.

Metric	Full KVAF	Geometry49	Binary Mask	EEF Pose Map
Visual Quality				
Image Quality	0.364	0.367	0.360	**0.376**
Aesthetic Quality	0.387	0.383	0.383	**0.389**
JEPA Similarity	**0.951**	0.830	0.939	0.932
Motion Quality				
Dynamic Degree	0.360	**0.361**	0.345	0.325
Flow Score	**0.273**	0.254	0.258	0.245
Motion Smoothness	0.730	**0.740**	0.725	0.690
Content Consistency				
Subject Consistency	**0.827**	0.804	0.812	0.822
Background Consistency	**0.904**	0.894	0.891	0.893
Photometric Consistency	0.220	0.231	**0.256**	0.248
Physics Adherence				
Interaction Quality	0.682	0.666	0.682	**0.696**
Trajectory Accuracy	**0.430**	0.230	0.357	0.282
3D Accuracy				
Depth Accuracy	**0.959**	0.847	0.932	0.892
Perspectivity	**0.838**	0.828	0.836	0.812
Controllability				
Instruction Following	**0.792**	0.758	0.780	0.716
Semantic Alignment	**0.895**	0.878	0.888	0.880
Action Following	0.0480	0.0476	0.0475	**0.0491**
EWMScore	**60.38**	56.99	59.32	57.79
P3CScore	**76.60**	70.11	74.60	71.30

Despite containing the same underlying robot and camera information, Geometry49 is substantially weaker than Full KVAF, most notably on Trajectory Accuracy (0.230 vs. 0.430) and Depth Accuracy (0.847 vs. 0.959), resulting in a 6.49-point P3CScore gap. The improvement is also concentrated in robot-centric geometry rather than generic visual quality: compared with Geometry49, Binary Mask changes Image Quality, Aesthetic Quality, and Background Consistency only marginally, while improving Trajectory Accuracy by 12.72 points and Depth Accuracy by 8.54 points. This suggests that camera-aligned spatial organization, rather than privileged geometric information alone, is critical for making robot motion and 3D structure easier for the video model to exploit.

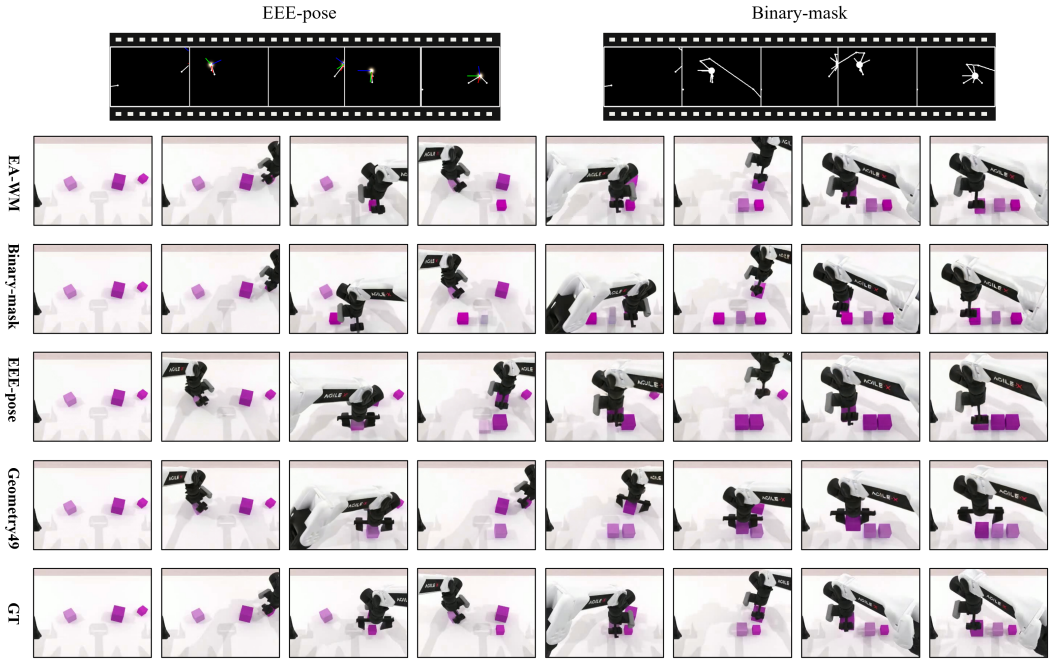

Figure 11: Qualitative comparison of simplified visual action representations and their corresponding RGB rollouts. EEF Pose Map provides sparse end-effector position and orientation cues, whereas Binary Embodiment Mask additionally exposes coarse full-arm geometry in the camera view.

The simplified visual controls show a consistent hierarchy. EEF Pose Map remains competitive on local Interaction Quality (0.696), but its sparse endpoint cues leave the global arm trajectory and depth structure under-constrained, yielding 0.282 Trajectory Accuracy and 0.892 Depth Accuracy. Binary Mask improves these metrics to 0.357 and 0.932 by explicitly constraining coarse full-arm occupancy, consistent with the qualitative rollouts in Fig. 11. Full KVAF further reaches 0.430 and 0.959 by retaining articulated topology, absolute camera-depth cues, joint locations, gripper state, manipulation focus, and end-effector pose. These results indicate that the advantage of KVAF comes from the structured kinematic content retained in the representation, rather than simply from additional model capacity or rasterizing robot motion into image space.

A.8 Evaluation Protocol and Baseline Reproduction

All EA-WM results, ablations, and locally reproduced baselines are evaluated on the released WorldArena validation set using the same evaluation pipeline and preprocessing. For external baselines, we use the results reported by WorldArena, which evaluates representative video and embodied world models under a unified benchmark protocol. We retain these reported values to maintain consistent provenance across the comparison tables. Since complete model-specific post-training configurations are not available for every baseline, these comparisons should be interpreted as benchmark-level comparisons under a common data split and evaluator, rather than strictly parameter- or optimization-matched comparisons.

As an additional sanity check, we locally post-trained both Wan2.2 and CtrlWorld on the released RoboTwin split using the same overall data and optimization budget as our main experiments, while retaining their original architectures and conditioning interfaces. The reproduced results closely match the corresponding WorldArena values, as summarized in Table 9. The small discrepancies observed for both a text-conditioned model and an action-conditioned model suggest that our local evaluation setup is well aligned with the benchmark protocol. We therefore use the WorldArena-reported numbers for external baselines in the main comparisons and use local reproductions primarily as protocol verification.

Table 9: Local verification of the WorldArena evaluation protocol. Values report EWMScore; the reproduced Wan2.2 and CtrlWorld results closely match the corresponding WorldArena values.

Model	WorldArena	Local reproduction	Abs. difference
Wan2.2	54.55	54.62	0.07
CtrlWorld	59.70	58.97	0.73

EA-WM is evaluated under two separately trained configurations, depending on whether future KVAFs are generated jointly with RGB video or supplied as action-derived conditions. In the main joint-generation setting, the model receives only the initial frame and text instruction; both future RGB and KVAF latents are initialized from noise and generated jointly. This setting is compared with text-conditioned world models. We additionally train a KVAF-conditioned variant for comparison with action- or trajectory-conditioned models. In this setting, the action trajectory is first converted into reference KVAFs, which are provided to the model as conditions for RGB generation, while the KVAF prediction head is removed.

Table 10: Two evaluation configurations of EA-WM. The joint-generation setting evaluates future RGB–KVAF co-generation from an initial observation and text instruction, whereas the KVAF-conditioned setting evaluates KVAFs as an explicit action-conditioning interface.

Setting	Inference input	KVAF role	Comparison
Joint-generation EA-WM	Initial frame + text	Future RGB and KVAF latents are jointly generated from noise	Text-conditioned models
KVAF-conditioned EA-WM	Initial frame + text + action-derived KVAFs	Reference KVAFs condition RGB generation	Action-/trajectory-conditioned models

This distinction is important when interpreting the two comparison groups: the joint-generation setting evaluates whether EA-WM can jointly model future visual and kinematic evolution from an initial observation and language instruction, whereas the KVAF-conditioned setting evaluates the effectiveness of KVAFs as an explicit action-conditioning interface. The latter therefore has access to a more informative action-derived condition and is not intended as a directly matched replacement for the joint-generation setting. Together, the two evaluations characterize EA-WM both as a generative world model and as an action-conditioned video predictor.

A.9 Computational Overhead

EA-WM introduces a second full-depth DiT stream for modeling KVAFs, and therefore has a larger computational footprint than the original single-stream Wan2.2-TI2V-5B backbone. We quantify this overhead in terms of model size, inference latency, GPU memory usage, and KVAF construction cost.

Model size and inference overhead. We benchmark Wan2.2-TI2V-5B and EA-WM on a single NVIDIA A100-SXM4-80GB GPU using bfloat16, 480×640 resolution, and 50 denoising steps. All latency results are averaged over three runs after warm-up and include preprocessing, denoising, and RGB VAE decoding.

EA-WM contains 10.40B DiT-side parameters, compared with 5.00B for Wan2.2-TI2V-5B. Most of the additional parameters come from the full-depth KVAF branch. For RGB generation, inference latency increases from 49.30 s to 96.38 s, corresponding to an increase of 47.08 s (95.5%), while

Table 11: Computational overhead of EA-WM relative to Wan2.2-TI2V-5B. DiT-side parameter counts exclude the shared video VAE and text encoder.

Method	DiT-side Params.	Latency (s)	Peak Memory (GiB)
Wan2.2-TI2V-5B	5.00B	49.30 ± 0.08	24.79
EA-WM (RGB)	10.40B	96.38 ± 0.05	34.88
EA-WM (RGB + KVAF decoding)	10.40B	100.89 ± 0.07	34.88

peak allocated GPU memory increases from 24.79 GiB to 34.88 GiB, an increase of 10.09 GiB (40.7%). We therefore do not claim computational or parameter parity with the single-stream Wan2.2 backbone. Decoding the generated KVAF visualization is optional and adds only 4.51 s without further increasing peak GPU memory.

Importantly, the architectural overhead above should be distinguished from the contribution of the KVAF representation itself. In the capacity-controlled representation analysis in Sec. A.7, we keep the dual-stream architecture, fusion design, training configuration, and parameter budget fixed while changing only the visual action representation. These controlled experiments show that the performance improvement cannot be explained solely by the increased model capacity.

KVAF construction cost. We separately benchmark KVAF construction on an Intel Xeon Platinum 8362 CPU. The measurement includes HDF5 loading, forward-kinematic projection, depth preprocessing, visual-field rendering, and video encoding. The measured processing times are 0.113 s, 0.401 s, 0.722 s, and 0.879 s for trajectories of 9, 33, 65, and 81 frames, respectively.

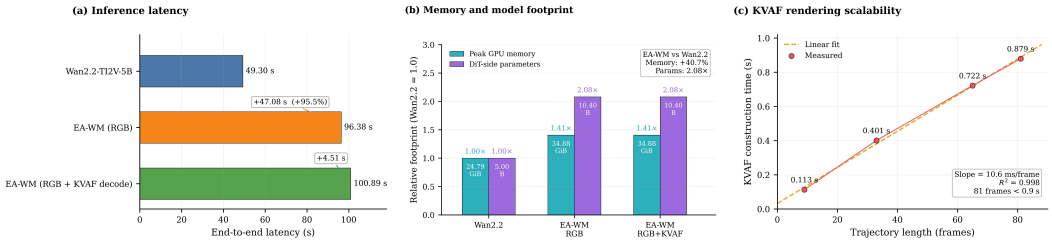

Figure 12: Computational profile of EA-WM. **(a)** End-to-end inference latency under the same sampling configuration. EA-WM introduces a 95.5% latency increase over Wan2.2-TI2V-5B, while optional KVAF decoding adds only 4.51 s. **(b)** Peak allocated GPU memory and DiT-side parameter footprint. Most of the additional computational footprint comes from the full-depth KVAF stream. **(c)** CPU cost of KVAF construction as trajectory length increases. Rendering scales approximately linearly at 10.6 ms per frame ($R^2 = 0.998$), with an 81-frame sequence requiring 0.879 s.

As shown in Fig. 12(c), KVAF construction scales approximately linearly with trajectory length, at about 10.6 ms per frame. Even an 81-frame trajectory requires less than 0.9 s to construct, indicating that KVAF rendering itself accounts for only a small fraction of the end-to-end computational cost. The dominant overhead of the current implementation instead comes from iterative denoising through the additional full-depth DiT stream.

Moreover, in the main joint-generation setting, no reference future KVAF trajectory is required at inference: both future RGB and KVAF latents are initialized from noise and jointly generated. Therefore, reference KVAF construction does not introduce an additional online conditioning cost in the main EA-WM setting.

Overall, the current implementation incurs substantial overhead primarily because of the full-depth dual-stream architecture, rather than the construction of KVAFs themselves. At the same time, the capacity-controlled experiments in Sec. A.7 separate this architectural cost from the benefit of the structured KVAF representation. Reducing the computational footprint through a shallower or weight-shared KVAF branch, fewer fusion layers, or model distillation is an important direction for future work.

A.10 Functional Utility of EA-WM

Beyond perceptual evaluation, we further examine whether the generated KVAF and RGB rollouts can support downstream robot learning, policy evaluation, and action planning.

Data Engine. Following the WorldArena data-engine setting, we evaluate on `adjust_bottle` and `click_bell`. We use a lightweight temporal Transformer to decode generated KVAF sequences into robot actions. Each 81-frame RGB KVAF sequence is encoded frame-wise by a small convolutional encoder into 128-dimensional tokens, followed by a 4-layer, 4-head Transformer encoder and an MLP head that predicts an 80-step, 14-dimensional action sequence. The decoder is trained on ground-truth KVAF–action pairs and kept fixed for downstream evaluation. The decoded trajectories are subsequently used to fine-tune the $\pi_{0.5}$ policy.

Table 12: Data-engine evaluation on the two WorldArena tasks. Actions decoded from generated KVAF trajectories provide effective supervision for downstream policy learning.

Training data	`adjust_bottle`	`click_bell`	Average
Zero-shot $\pi_{0.5}$	2%	5%	3.5%
WoW	45%	71%	58.0%
GT RoboTwin data	77%	66%	71.5%
Generated KVAF $\rightarrow$ actions	75%	69%	**72.0%**

Actions decoded from generated KVAFs achieve 72.0% average success, compared with 58.0% for the strongest reported world-model data engine and 71.5% for the ground-truth RoboTwin-data reference on these two tasks. This result suggests that the jointly generated KVAF stream retains substantial executable information and can serve as a synthetic interface for downstream policy learning.

Policy Evaluation. Following the WorldArena policy-evaluation methodology, we evaluate five $\pi_{0.5}$ policies trained with 10%, 20%, 30%, 50%, and 100% of the training data. For each policy, we perform two runs of the same 500 task–initialization cases (50 tasks, 10 cases per task), reusing the same seed configuration. We report the arithmetic mean of the two run-level success rates, separately for RoboTwin execution and EA-WM-plus-VLM evaluation, and compute Pearson and Spearman correlations across the five pairs of averaged success rates. These are repeated measurements on a fixed evaluation set, not independent-seed evaluations.

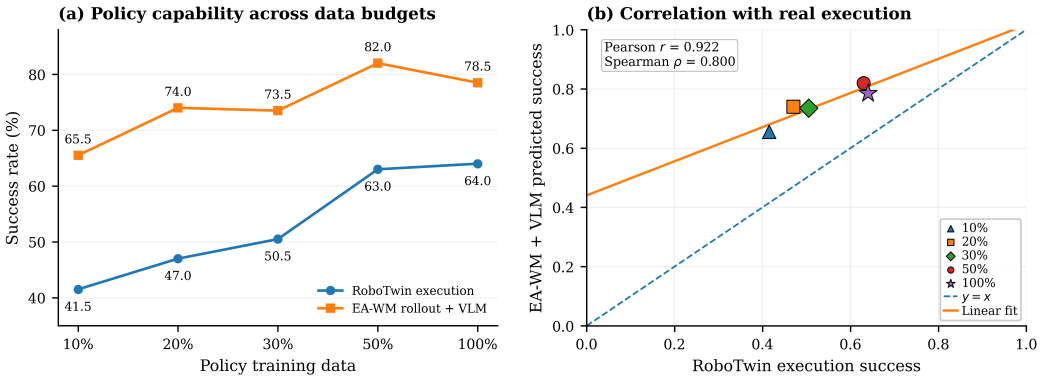

Figure 13: Policy evaluation with EA-WM. **(a)** RoboTwin execution success and EA-WM rollout-based evaluation for policies trained with different data budgets. **(b)** Correlation between RoboTwin execution success and EA-WM rollout evaluation across the five policies. EA-WM reaches Pearson $r = 0.922$ and Spearman $\rho = 0.800$. Each point reports the mean success rate over two repeated runs of the same 500 evaluation cases.

The rollout evaluator is somewhat optimistic in absolute success rate, but tracks policy capability consistently as the training-data budget changes. Its Pearson correlation is close to the strongest result reported by WorldArena (CtrlWorld, $r = 0.986$) and substantially higher than Cosmos-Predict

2.5 ($r = 0.483$). These results indicate that EA-WM rollouts provide a useful proxy for comparing policies with different capability levels.

Action Planning. Finally, we use the temporal decoder to convert generated KVAF trajectories into executable actions and evaluate them directly in RoboTwin under the WorldArena action-planning protocol.

Table 13: Action-planning success on the two WorldArena tasks. EA-WM achieves the strongest average success among the compared world-model planners.

Planner	`adjust_bottle`	`click_bell`	Average
$\pi_{0.5}$ policy	77%	66%	71.5%
WoW	20%	21%	20.5%
TesserAct	1%	35%	18.0%
EA-WM KVAF planning	35%	**97%**	**66.0%**

EA-WM reaches 66.0% average planning success, substantially outperforming the other world-model planners in this comparison and approaching the direct $\pi_{0.5}$ policy reference. Performance is particularly strong on `click_bell` (97%), while the lower success on `adjust_bottle` (35%) indicates that precise multi-stage manipulation remains challenging. This gap is consistent with the fact that planning depends on both the accuracy of generated KVAF trajectories and the subsequent KVAF-to-action decoding process.

A.11 Preliminary Real-World Evaluation on RH20T

We further conduct a preliminary offline evaluation on RH20T to examine whether EA-WM remains effective beyond the simulated RoboTwin setting. We use data collected with a UR5 robot and WSG-50 gripper from a fixed global camera, which introduces both real-world appearance and a different robot embodiment from our main experiments. We construct 2,000 short clips for post-training and evaluate on 10 held-out episodes covering 10 tasks. Table 14 reports the six robot-centric metrics used by P3CScore together with the overall aggregate score.

Table 14: Preliminary real-world evaluation on RH20T. EA-WM achieves the best overall P3CScore, with the largest gains concentrated on interaction, trajectory, and depth modeling.

	Physics Adherence		3D Accuracy		Controllability			
Model	Inter.	Traj.	Depth	Persp.	Instr.	Sem.	P3CScore	EWMScore
CtrlWorld (w/o PT)	0.400	0.279	0.802	0.600	0.276	0.831	53.12	47.97
Wan2.2 (w/o PT)	0.362	0.308	0.801	0.738	0.244	0.840	54.88	51.94
CtrlWorld (PT)	0.468	0.507	0.889	0.798	0.342	0.867	64.51	54.41
Wan2.2 (PT)	0.508	0.563	0.849	**0.858**	**0.378**	**0.877**	67.22	56.51
EA-WM (PT)	**0.542**	**0.608**	**0.927**	0.808	0.374	0.866	**68.75**	**57.59**

EA-WM achieves the best P3CScore of 68.75 and the highest overall EWMScore of 57.59. Compared with the strongest post-trained baseline, Wan2.2, the improvement is most pronounced on the metrics directly targeted by KVAF: Interaction Quality increases from 0.508 to 0.542, Trajectory Accuracy from 0.563 to 0.608, and Depth Accuracy from 0.849 to 0.927. EA-WM does not improve every individual metric, but its gains are concentrated on robot motion and 3D geometry, consistent with the trends observed in the RoboTwin experiments.

The qualitative examples in Fig. 14 show a similar trend. EA-WM can preserve the overall scene layout while generating plausible robot motion and interaction progress across held-out real-world tasks, although occasional appearance and geometry drift remain. Since this evaluation is limited to one robot embodiment, one fixed camera, 2,000 post-training clips, and 10 held-out episodes, we view these results as preliminary evidence of real-world transfer rather than a comprehensive evaluation of robustness or embodiment generalization.

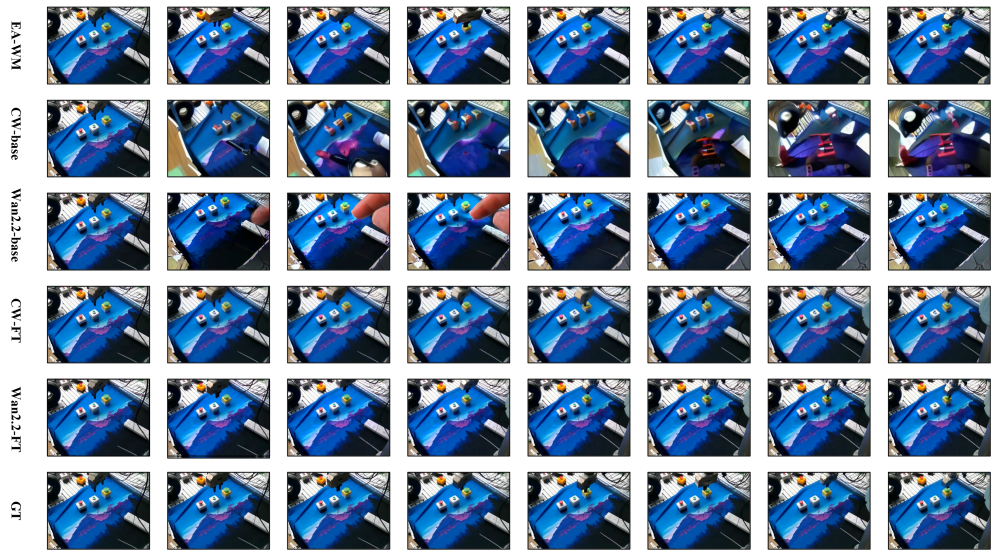

Figure 14: Qualitative rollouts on held-out RH20T episodes. The examples illustrate that EA-WM can preserve the overall real-world scene layout while generating plausible robot motion and interaction progress across different tasks. Occasional appearance and geometry drift remain in some rollouts, indicating that real-world generation is still imperfect.

A.12 Counterfactual Condition Following

We further examine whether EA-WM generates futures that respond to the supplied conditions rather than relying primarily on the initial observation. We construct a counterfactual test in which the initial frame is kept fixed, while both the text instruction and KVAF trajectory are replaced by those from a different task. Specifically, we retain the initial observation from `beat_block_hammer` and replace its original condition with the instruction and KVAF trajectory from `adjust_bottle`.

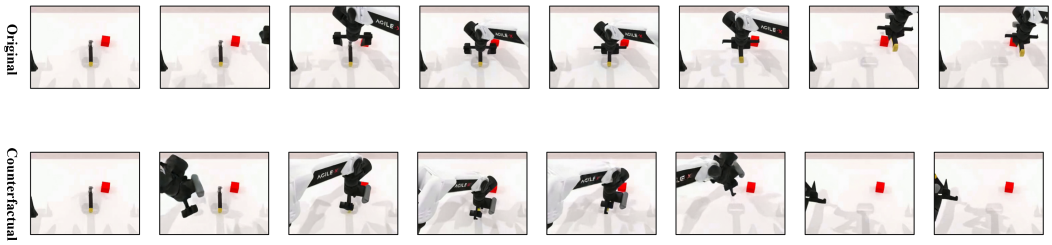

Figure 15: Counterfactual condition-following results. Both rollouts start from the same initial observation. The top row uses the original condition (`beat_block_hammer`), while the bottom row uses the counterfactual condition (`adjust_bottle`). Changing the text and KVAF conditions leads to clearly different robot trajectories while preserving the initial scene context.

As shown in Fig. 15, the two rollouts share the same initial scene but quickly diverge in their subsequent robot motion. Under the original condition, the arm follows a trajectory concentrated around the original manipulation region, whereas the counterfactual condition induces a distinct approach and motion pattern. Importantly, the surrounding scene and object layout remain largely consistent while the robot trajectory changes, consistent with the model preserving scene context while responding to condition-dependent motion cues. This result provides qualitative evidence that EA-WM does not simply reproduce a canonical continuation associated with the first frame, but uses the supplied text and KVAF trajectory to modulate future robot motion and generate condition-dependent rollouts.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the key contribution and scope of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in Section 5 of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not present formal theoretical results. The method design, model architecture, and implementation details are described in Section 3 and Appendix A.1, Appendix A.2, and Appendix A.3.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The model architecture, KVAF construction, and training objective are in Section 3 and Appendices A.1–A.3. Section 4.1 describes the training setup, and Appendix A.8 specifies the evaluation protocol, baseline-result provenance, local baseline verification, and the two separately trained EA-WM configurations. Appendices A.10 and A.11 describe the downstream evaluation and preliminary real-world evaluation, respectively.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide the code and dataset in the appendix materials.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and test details are provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars or statistical significance tests because repeated full training and evaluation runs would incur substantial computational cost. We compare all models under the same WorldArena evaluation protocol and report the corresponding results in Table 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information about the compute resources needed for reproduction is provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: A discussion of the broader societal impacts is provided in the Appendix A.6.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not pose such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the public RoboTwin2.0 dataset/platform and the open-source Wan2.2 model. These are properly cited in the paper and accompanied by the corresponding references. These assets are widely used in the community, and their licenses follow the terms provided by the original sources.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code, inference results, and model weights introduced in the paper are all accompanied by detailed documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve any human subjects or crowdsourcing experiments. All results are obtained through objective evaluation on public datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve any human subjects or crowdsourcing, and therefore does not require IRB approval.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development of this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.