ACTIONS AS RAYS: REPURPOSING VIDEO DIFFUSION MODELS FOR ROBOT CONTROL WITH VISUAL ACTION REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained video diffusion models have learned rich priors over visual dynamics and physical interactions, yet adapting them to robot manipulation remains challenging, as existing approaches often rely on dedicated action heads, specialized tokenizers, or auxiliary policy networks to translate between visual observations and low-dimensional action signals. In this work, we introduce RayWAM, a World-Action Model that addresses this challenge through Action Raymap, a video-aligned and overcomplete action representation that transforms robot actions into structured RGB maps. By interpreting each end-effector action as the pose of a virtual camera, Action Raymap represents translation, rotation, and gripper states as dense spatial fields, while maintaining a deterministic mapping back to executable robot commands. Based on this formulation, observations and actions are represented in a unified RGB space and encoded using the same frozen pretrained video VAE, projecting robot actions directly into the latent space of pretrained video diffusion models. Unlike prior approaches that adapt video models through additional action-specific modules, RayWAM instead adapts the action representation itself to match the native modality of the pretrained model, enabling more effective exploitation of transferable representations and physical dynamics priors learned from large-scale video data. We evaluate RayWAM on LIBERO, LIBERO-Plus, and VLABench, together with real-world robotic experiments, demonstrating consistent improvements in task success rates, robustness to environmental variations, and real-world generalization across diverse manipulation scenarios.

1 INTRODUCTION

Learning robot manipulation policies from pretrained video models offers a way to transfer visual knowledge of motion and interaction to control. World-Action Models (WAMs) pursue this direction by coupling predictive visual dynamics with action generation (Ye et al., 2026; Li et al., 2026; Yuan et al., 2026b). However, video models are pretrained to process visual observations, whereas robot actions are typically expressed as low-dimensional continuous signals. To bridge this gap, many existing approaches introduce dedicated action experts or other action-specific modules (Yuan et al., 2026b; Chen et al., 2026a), but such designs do not fully exploit the transferable representations and dynamics priors learned during video pretraining. A key challenge is therefore *how to represent robot actions in a form that is compatible with the native visual space of pretrained video models*.

Recent work has explored representing robot actions directly in visual space to bridge this modality gap. However, existing formulations either rely on calibrated real-camera parameters to project or render actions into observation views (Jiang et al., 2025; Chen et al., 2026b; Wang et al., 2025; Zhen et al., 2026), or model visual motion rather than explicit actions, requiring an additional mapping back to robot actions (Alzayer et al., 2026; Chen et al., 2026a). These limitations motivate a visual action representation that maps executable actions into the native visual space of pretrained video models, without relying on real-camera parameters or additional action decoding modules.

In this paper, we introduce **RayWAM**, which realizes this principle through a video-aligned, overcomplete representation of executable robot actions (Fig. 1). Our key insight is that robot actions can

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

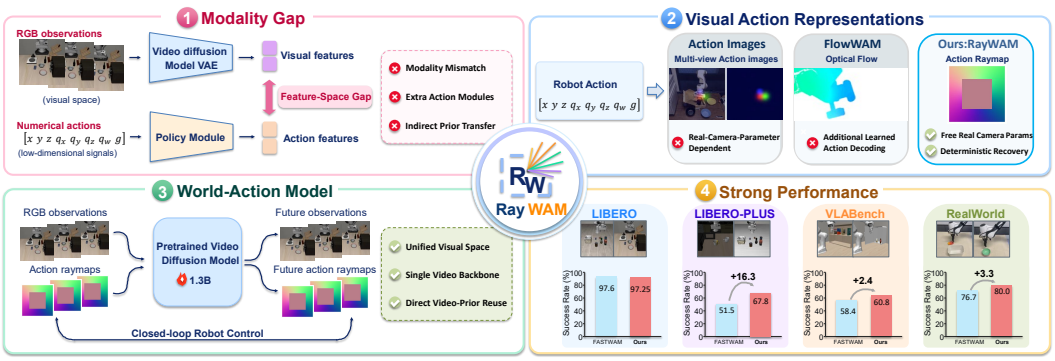

Figure 1: **Overview of RayWAM.** Action Raymap represents executable robot actions as structured RGB maps without requiring real-camera parameters while remaining deterministically recoverable. This enables joint observation–action modeling with a pretrained video diffusion backbone.

be transformed into a visually compatible form by mapping end-effector poses into RGB-formatted frames. Specifically, we interpret each end-effector pose as the pose of a virtual camera whose coordinate frame is aligned with the end-effector frame. The virtual camera intrinsics are fixed by design, requiring no camera calibration or observation-frame transformation. Based on this formulation, we develop **Action Raymap**, which represents translation, rotation, and gripper state as structured RGB maps. Action Raymap distributes action information over a dense spatial field, yielding an overcomplete representation with structured redundancy, where spatial elements are deterministically coupled by the underlying robot pose. Such consistency cues provide structured correlations that can be exploited by video models during action generation.

With Action Raymap, we formulate robot policy learning as joint video-action generation, treating robot actions as video frames. RGB observations and Action Raymaps are encoded by the same frozen pretrained video VAE encoder into a shared visual latent space, thereby aligning action representations with the visual representations learned during large-scale video pretraining. The pretrained video diffusion backbone is then fine-tuned to jointly generate future observations and Action Raymaps. To preserve the dependency of action generation on visual dynamics, we introduce asymmetric block-wise attention. The generated Action Raymaps are deterministically decoded into executable robot actions, enabling closed-loop control without additional action prediction heads or auxiliary policy networks.

We evaluate RayWAM on three simulation benchmarks: LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2025), and VLABench (Zhang et al., 2025), covering general manipulation, out-of-distribution generalization, and progress-aware manipulation. The results demonstrate that representing actions as videos enables effective robot control using a pretrained video diffusion model. We further evaluate RayWAM on real-world manipulation tasks to assess its applicability beyond simulation. Our main contributions are:

1. We introduce **Action Raymap**, a video-aligned, overcomplete visual representation of end-effector actions that supports deterministic recovery without real-camera parameters.
2. We develop **RayWAM**, which repurposes a pretrained video diffusion model to jointly generate future observations and actions without introducing any action-specific modules or requiring architectural modifications, thereby preserving the capabilities inherited from large-scale pretraining.
3. We extensively evaluate RayWAM on simulation benchmarks and real-world manipulation tasks, showing that it achieves competitive manipulation performance while generalizing effectively to diverse environments and tasks.

2 Related Work

2.1 World-Action Models for Robot Manipulation

World-Action Models (WAMs) leverage predictive dynamics from pretrained video models for robot control, broadly following three paradigms. The first line of work generates future visual observa-

tions before inferring executable actions, as in LingBot-VA (Li et al., 2026) and Vidar (Feng et al., 2025); the second predicts actions from intermediate denoising features or latent predictive representations (Liang et al., 2025; Ma et al., 2026; Yuan et al., 2026b; Sun et al., 2026; Shen et al., 2026); and the third jointly models future visual dynamics and robot actions within pretrained video architectures (Ye et al., 2026; Bi et al., 2026; Wang et al., 2026c; Team et al., 2026; Zhang et al., 2026; Wu et al., 2026). Recent work further enriches predictive representations with motion, geometry, semantics, or task progress to improve their utility for control (Routray et al., 2026; Yan et al., 2026; Yuan et al., 2026a).

Across these paradigms, action representation remains a key interface between video dynamics and executable control. Cosmos Policy (Kim et al., 2026) models numerical actions directly in the latent sequence of a pretrained video diffusion model, while RepWAM (Wang et al., 2026a) learns aligned visual–action representations. Our work instead asks whether executable robot actions can be represented directly in the native visual space of pretrained video models.

2.2 Visual Action Representations for Robot Learning

Recent work increasingly represents robot control in visual space to better match the native modality of pretrained video models. One line projects or renders action-related geometry into camera-aligned visual representations, including projected action maps, embodiment masks, visual prompts, and multi-view action images (Jiang et al., 2025; Chen et al., 2026b; Wang et al., 2025; Zhen et al., 2026); although spatially grounded, these representations remain tied to real-camera parameters for projection or rendering. Another line represents action-relevant dynamics through visual motion, such as masked visual trajectories and dense optical-flow videos (Alzayer et al., 2026; Liu et al., 2026; Chen et al., 2026a). FlowWAM (Chen et al., 2026a), for example, uses optical flow as a video-aligned action representation but still relies on a learned action expert to recover executable commands. These limitations motivate a visual action representation that reparameterizes executable actions while remaining compatible with the native visual space of pretrained video models.

2.3 Ray-based representations

Ray-based representations provide a natural basis for constructing such a visual interface, as they expand compact 3D poses into structured spatial signals. Cameras as Rays (Zhang et al., 2024) replaces global camera poses with dense pixel-associated ray bundles, while camera-controlled video generation methods encode camera trajectories through spatial ray or Plücker representations (He et al., 2025a; Xu et al., 2024; Zheng et al., 2024; Bahmani et al., 2025; He et al., 2025b). More closely related to our formulation, Rays as Pixels (Jang et al., 2026) converts camera trajectories into three-channel ray images that can be processed by the same pretrained video VAE as RGB frames, and GenCeption (Wang et al., 2026b) introduces the "Rothko" representation to reorganize a six-channel raymap into a standard three-channel image. Building on this representation principle, we interpret each end-effector pose as a virtual-camera pose and extend ray-based representations from camera trajectories to executable robot actions, yielding a real camera-parameter-free formulation with deterministic action recovery.

3 Method

We present **RayWAM**, a World-Action Model that preserves the native architecture of a pretrained video diffusion model by representing robot actions directly in the visual domain. This formulation allows visual observations and robot actions to share the same video VAE and video diffusion backbone. The overall framework is illustrated in Fig. 2.

3.1 Problem Formulation and Notation

We formulate robot manipulation in an end-effector control space, where the current end-effector state and each action are parameterized by a position $\mathbf{p} \in \mathbb{R}^3$, a unit quaternion $\mathbf{q}$, and a binary gripper state $g \in \{0, 1\}$. We denote the current end-effector state by $\mathbf{s}_t = (\mathbf{p}_t, \mathbf{q}_t, g_t)$, where $\mathbf{p}_t$ and $\mathbf{q}_t$ specify the absolute end-effector pose in a fixed robot base coordinate frame. An action is denoted by $\mathbf{a}_t = (\mathbf{p}_t^a, \mathbf{q}_t^a, g_t^a)$, where $\mathbf{p}_t^a$ and $\mathbf{q}_t^a$ represent the target translation and rotation relative

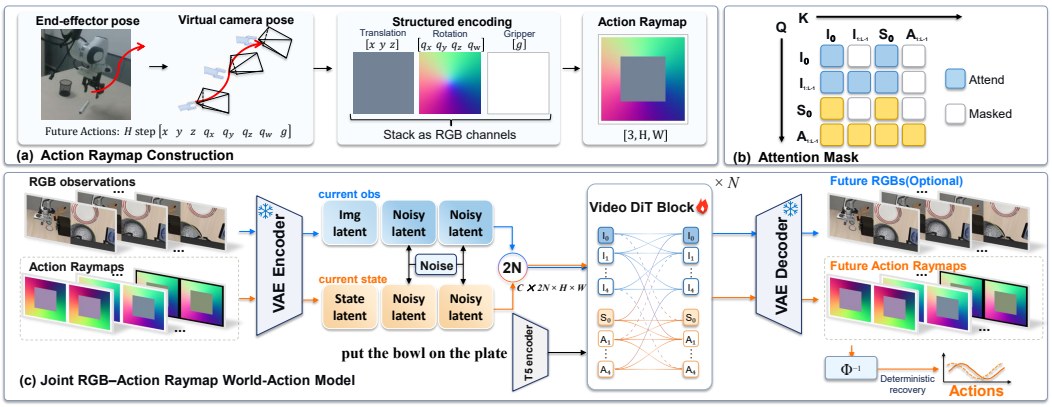

Figure 2: **Architecture of RayWAM.** Robot actions are represented directly in the visual domain, enabling observations and actions to share the pretrained video VAE and video diffusion backbone.

to the current state $\mathbf{s}_t$, computed using the standard SE(3) transformation. All actions within a predicted chunk share the same current state as their reference. Given a language instruction ℓ, the current multi-view observation $\mathbf{o}_t$ (Appendix A.2), and the current end-effector state $\mathbf{s}_t$, our goal is to jointly predict future visual observations $\mathbf{o}_{t+1:t+H}$ and an H-step action chunk $\mathcal{A}_t = \mathbf{a}_{t:t+H-1}$.

We form a sequence by prepending the current end-effector state to the action chunk:

$$\mathcal{X}_t = \{(\mathbf{p}_t^{(k)}, \mathbf{q}_t^{(k)}, g_t^{(k)})\}_{k=0}^{H} = (\mathbf{s}_t, \mathcal{A}_t), \tag{1}$$

where $(\mathbf{p}_t^{(0)}, \mathbf{q}_t^{(0)}, g_t^{(0)}) = \mathbf{s}_t$ represents the absolute current end-effector state, and the remaining H elements correspond, in temporal order, to future actions expressed relative to the same state. We convert $\mathcal{X}_t$ into a sequence of $(H+1)$ three-channel Action Raymaps:

$$\mathcal{M}_t = \Phi(\mathcal{X}_t) = \mathbf{m}_{t:t+H}, \tag{2}$$

where $\mathbf{m}_t$ represents the current absolute end-effector state and $\mathbf{m}_{t+1:t+H}$ represent the corresponding relative future actions.

The resulting representation allows the WAM to model the conditional joint distribution

$$p_\theta(\mathbf{o}_{t+1:t+H}, \mathbf{m}_{t+1:t+H} \mid \mathbf{o}_t, \mathbf{m}_t, \ell), \tag{3}$$

over future visual observations and future Action Raymaps. At inference time, the predicted Action Raymaps are deterministically mapped back to the corresponding numerical action chunk:

$$\hat{\mathcal{A}}_t = \Phi^{-1}(\hat{\mathbf{m}}_{t+1:t+H}) = \hat{\mathbf{a}}_{t:t+H-1}. \tag{4}$$

3.2 Action Raymap: Video-Aligned Overcomplete Action Representation

Action Raymap Construction. Following ray-based geometric formulations (Zhang et al., 2024; Wang et al., 2026b), we map the end-effector state and actions into structured visual representations by interpreting each element of $\mathcal{X}_t$ as the pose of a *virtual camera*. Because this virtual camera is an algorithmic abstraction, its viewing grid and intrinsic matrix can be fixed by design without physical camera calibration. Under pinhole geometry, each discrete pixel $\mathbf{u} = (u, v)$ on the virtual image plane defines a viewing ray originating from the optical center.

For the k-th element of $\mathcal{X}_t$, we denote its translation by $\mathbf{T}_t^{(k)} = \mathbf{p}_t^{(k)} \in \mathbb{R}^3$ and its rotation matrix by $\mathbf{R}_t^{(k)} = \mathrm{R}(\mathbf{q}_t^{(k)}) \in \mathrm{SO}(3)$. For $k = 0$, these quantities represent the absolute current end-effector pose, while for $k = 1, \ldots, H$, they represent the corresponding future pose relative to the current state. We unproject pixel $\mathbf{u}$ using a predefined intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ (Appendix A.7) and homogeneous coordinates $\tilde{\mathbf{u}} = [u, v, 1]^\top$ to obtain the canonical unit ray direction:

$$\mathbf{d}_\mathbf{u} = \frac{\mathbf{K}^{-1}\tilde{\mathbf{u}}}{\|\mathbf{K}^{-1}\tilde{\mathbf{u}}\|_2} \in \mathbb{R}^3. \tag{5}$$

The corresponding ray origin is given by $\mathbf{T}_t^{(k)}$, and its direction is transformed as $\mathbf{R}_t^{(k)}\mathbf{d}_\mathbf{u}$.

Stacking the 3D ray origin and transformed direction at each pixel yields a conventional 6-channel raymap, which is incompatible with off-the-shelf 3-channel video VAE encoders. We spatially partition these components into a unified 3-channel **Action Raymap** (Fig. 2(a)): the central region $\mathcal{C}$ is tiled with the translation vector $\mathbf{T}_t^{(k)}$, the surrounding region $\mathcal{D}$ contains the transformed ray field $\mathbf{R}_t^{(k)}\mathbf{d_u}$, and the outer boundary $\mathcal{B}$ represents the binary gripper state $g_t^{(k)}$. Prior to VAE tokenization, translation coordinates are normalized to $[-1, 1]$, using separate statistics for the absolute current state and the relative future actions.

Deterministic Action Recovery. The overcomplete structure of Action Raymap enables deterministic action recovery without auxiliary learned policy heads. In our multi-view implementation, the horizontally replicated Action Raymap tiles are first averaged pixel-wise after VAE decoding, and action recovery is performed once on the resulting averaged map. For a predicted future map $\hat{\mathbf{m}}_{t+k}$ $(k = 1, \ldots, H)$, the relative translation $\hat{\mathbf{p}}^a_{t+k-1}$ is recovered from the channel-wise median of the central region $\mathcal{C}$, followed by inverse rescaling using the relative-action normalization statistics, and the discrete gripper state $\hat{g}_t^{(k)}$ is extracted from the boundary band $\mathcal{B}$. To recover the relative rotation, we estimate the rotation matrix that aligns the canonical ray directions with the predicted ray field across region $\mathcal{D}$:

$$\widehat{\mathbf{R}}_t^{(k)} = \arg\min_{\mathbf{R}\in \mathrm{SO}(3)} \sum_{\mathbf{u}\in\mathcal{D}} \left\| \hat{\mathbf{d}}^{(k)}_{t,\mathbf{u}} - \mathbf{R}\mathbf{d}_{\mathbf{u}} \right\|_2^2, \tag{6}$$

which admits a closed-form solution via SVD, as detailed in Appendix A.5. Converting $\widehat{\mathbf{R}}_t^{(k)}$ into a unit quaternion $\hat{\mathbf{q}}^a_{t+k-1}$, together with the recovered translation and gripper state, yields the relative action:

$$\hat{\mathbf{a}}_{t+k-1} = \left(\hat{\mathbf{p}}^a_{t+k-1}, \hat{\mathbf{q}}^a_{t+k-1}, \hat{g}_t^{(k)} \right). \tag{7}$$

Applying this mapping independently to $\hat{\mathbf{m}}_{t+1:t+H}$ recovers the full relative action chunk $\hat{\mathcal{A}}_t$.

3.3 World-Action Generation with a Native Video Diffusion Model

By representing robot actions as Action Raymaps, visual observations and robot actions share the same image-space interface. This allows us to repurpose a pretrained Wan2.1-1.3B video diffusion model (Wan et al., 2025) for joint world-action modeling without introducing action-specific tokenizers, prediction heads, or auxiliary policy networks. As illustrated in Fig. 2(c), both modalities are encoded by the same video VAE encoder and modeled by the same pretrained video DiT, largely preserving the native video diffusion architecture.

Shared Video Latent Sequence. Let $\mathcal{O}_t = [\mathbf{o}_t, \mathbf{o}_{t+1:t+H}]$ denote the visual observation sequence and $\mathcal{M}_t = [\mathbf{m}_t, \mathbf{m}_{t+1:t+H}]$ the corresponding Action Raymap sequence, where $\mathbf{m}_t$ represents the absolute current end-effector state and $\mathbf{m}_{t+1:t+H}$ represent future actions relative to this state. We independently encode both sequences using the shared, frozen pretrained video VAE encoder $\mathcal{E}$ and concatenate their latent representations along the temporal dimension:

$$\mathcal{Z}_t^o = \mathcal{E}(\mathcal{O}_t), \qquad \mathcal{Z}_t^m = \mathcal{E}(\mathcal{M}_t), \qquad \mathcal{Z}_t = [\mathcal{Z}_t^o \| _T \mathcal{Z}_t^m] = [\mathbf{z}_0^o, \ldots, \mathbf{z}_{L-1}^o \mid \mathbf{z}_0^m, \ldots, \mathbf{z}_{L-1}^m], \tag{8}$$

where L denotes the number of latent frames for each modality. For an action horizon of $H = 16$, each 17-frame RGB or Action Raymap sequence is compressed by the Wan2.1 VAE into $L = 5$ latent frames, forming a unified ten-frame joint latent sequence. Temporal concatenation preserves the latent spatial resolution and channel dimensions expected by the video backbone.

The initial latent tokens $\mathbf{z}_0^o$ and $\mathbf{z}_0^m$ (denoted as $\mathbf{I}_0$ and $\mathbf{S}_0$) serve as clean conditioning tokens for the current observation and robot state, respectively. Under the causal temporal encoding of the Wan VAE, the first latent frame depends only on the initial input frame, allowing these conditioning latents to be obtained from the current observation and state Raymap alone at inference time. The remaining future latent tokens $\mathbf{z}^o_{1:L-1}$ $(\mathbf{I}_{1:L-1})$ and $\mathbf{z}^m_{1:L-1}$ $(\mathbf{A}_{1:L-1})$ are jointly modeled by the video DiT conditioned on the language instruction ℓ.

Block-wise Attention. To jointly model visual dynamics and actions while maintaining a directional dependency between the two modalities, we apply a block-wise attention mask over

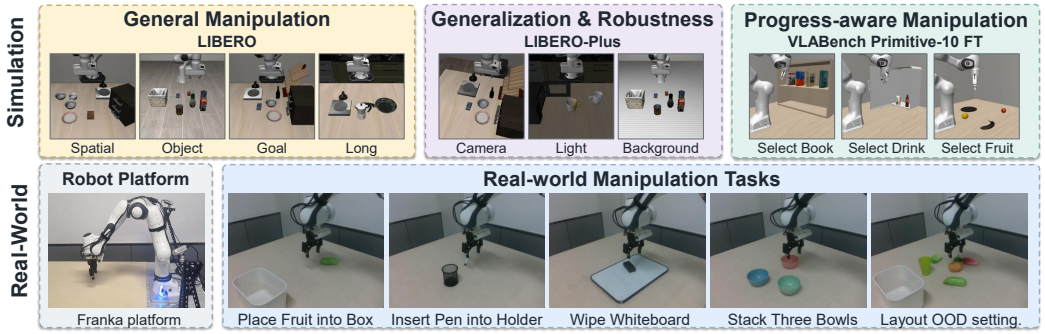

Figure 3: **Experimental evaluation settings of RayWAM.** We evaluate RayWAM on LIBERO, LIBERO-Plus, VLABench Primitive-10 FT, and four real-world manipulation tasks with an additional Layout OOD setting.

$[\mathbf{I}_0, \mathbf{I}_{1:L-1}, \mathbf{S}_0, \mathbf{A}_{1:L-1}]$, as shown in Fig. 2(b). Future visual latents $\mathbf{I}_{1:L-1}$ can attend to the current visual observation $\mathbf{I}_0$, other future visual latents $\mathbf{I}_{1:L-1}$, and the current robot state $\mathbf{S}_0$, while attention to future action latents $\mathbf{A}_{1:L-1}$ is masked. This allows the current robot state to inform visual dynamics without allowing future action targets to directly influence future visual prediction. Future action latents $\mathbf{A}_{1:L-1}$ can attend to the entire joint sequence, enabling action generation to leverage the predicted visual dynamics. The model jointly updates all future RGB and action latents at each diffusion step, with the attention mask controlling their cross-modal interaction.

Training Objective. Following Wan et al. (2025), we adopt the flow-matching objective for training. Let $\mathbf{Z}$ denote the clean joint latent sequence in Eq. (8) and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For a sampled noise level σ_τ, we interpolate:

$$\mathbf{Z}_\tau = (1 - \sigma_\tau)\mathbf{Z} + \sigma_\tau \boldsymbol{\epsilon}, \qquad \mathbf{v}^* = \boldsymbol{\epsilon} - \mathbf{Z}, \qquad \hat{\mathbf{v}} = f_\theta(\mathbf{Z}_\tau, \tau, \ell), \tag{9}$$

where the conditioning latents $\mathbf{z}_0^o$ and $\mathbf{z}_0^m$ remain uncorrupted throughout training. We compute the flow-matching velocity errors over the future RGB and Action Raymap blocks and optimize:

$$\mathcal{L}_{\text{policy}} = \lambda_{\text{rgb}} \mathbb{E}_{\tau,\boldsymbol{\epsilon}} \left[w(\tau) \left\| \hat{\mathbf{v}}_{\text{rgb}} - \mathbf{v}^*_{\text{rgb}} \right\|_2^2 \right] + \lambda_{\text{Action}} \mathbb{E}_{\tau,\boldsymbol{\epsilon}} \left[w(\tau) \left\| \hat{\mathbf{v}}_{\text{Action}} - \mathbf{v}^*_{\text{Action}} \right\|_2^2 \right], \tag{10}$$

where $w(\tau)$ denotes the native flow-matching timestep weighting function. We set λ_{rgb} and λ_{Action} to balance the loss magnitudes across modalities. Action supervision is applied entirely in the Action Raymap latent space, without auxiliary objectives on numerical action vectors. During policy training, the video DiT is fine-tuned while the VAE encoder remains frozen.

Closed-Loop Inference. During deployment, the current observation $\mathbf{o}_t$ and current-state Action Raymap $\mathbf{m}_t$ are independently encoded into the conditioning latents $\mathbf{z}_0^o$ and $\mathbf{z}_0^m$. The future RGB and Action Raymap latents are initialized from Gaussian noise and jointly denoised by the video DiT. Future RGB reconstruction is not required for control, so we decode only the predicted Action Raymap latents and recover the relative action chunk $\hat{\mathcal{A}}_t = \Phi^{-1}(\hat{\mathbf{m}}_{t+1:t+H})$ following Sec. 3.2.

4 Experiments

4.1 Simulation Experiments

Benchmarks and Baselines. We evaluate general manipulation on LIBERO (Liu et al., 2023), zero-shot generalization on LIBERO-Plus (Fei et al., 2025), and progress-aware manipulation on VLABench Primitive-10 FT (Zhang et al., 2025). We compare RayWAM with existing vision-language-action (VLA) policies and world-action models (WAMs). On LIBERO and LIBERO-Plus, we compare against UniVLA (Bu et al., 2025), OpenVLA-OFT (Kim et al., 2025), π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), $\pi_{0.5}$ (Intelligence et al., 2025), VLA-JEPA (Sun et al., 2026), Fast-WAM (Yuan et al., 2026b), and OpenWAM-α (Wang et al., 2026c). On VLABench, we additionally compare with Motus (Bi et al., 2026). An overview of the evaluation settings is shown in Fig. 3.

Table 1: **Results on LIBERO and LIBERO-Plus.** Success rates (%) are reported on LIBERO and seven LIBERO-Plus shifts. Per-suite performance of LIBERO is reported in Appendix B.1. Emb. PT. denotes additional embodied pretraining; best and second-best results are **bold** and underlined

				LIBERO	LIBERO-Plus Zero-Shot Transfer							
Method	Type	Params	Emb. PT.	Avg.	Camera	Robot	Lang.	Light	Bkg.	Noise	Layout	Avg.
UniVLA	VLA	7B	Yes	95.2	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
OpenVLA-OFT	VLA	7B	Yes	97.1	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π_0	VLA	3.3B	Yes	94.2	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
π_0-FAST	VLA	3.3B	Yes	85.5	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
$\pi_{0.5}$	VLA	3.3B	Yes	96.9	**78.4**	73.6	80.8	96.2	**94.1**	**89.0**	84.5	**84.4**
VLA-JEPA	WAM	2B	Yes	97.2	63.3	67.1	85.4	95.6	93.6	66.3	**85.1**	79.5
w/o human videos	WAM	2B	Yes	96.1	40.3	55.7	72.9	88.2	70.5	38.2	74.6	62.9
Fast-WAM	WAM	6B	No	97.6	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
OpenWAM-α	WAM	6B	Yes	**99.3**	33.8	**76.1**	88.0	**97.0**	87.1	39.8	77.5	69.2
Ours	WAM	**1.3B**	No	97.25	70.36	45.81	**90.11**	89.75	48.61	48.03	83.08	67.84

Implementation Details. We initialize RayWAM from the pretrained Wan2.1-T2V-1.3B model Wan et al. (2025) and use an action horizon of $H = 16$. We fully fine-tune the video DiT while keeping the VAE encoder frozen during policy training. For Action Raymap reconstruction, we adapt only the VAE decoder while keeping the encoder fixed; its effect is evaluated in Sec. 4.3. We use AdamW with a learning rate of 1×10^{-4} and an effective batch size of 128, and use 20 denoising steps at inference. Additional details are provided in Appendix A.

General Manipulation on LIBERO. RayWAM achieves competitive general manipulation performance on LIBERO using a substantially smaller backbone and no additional embodied pretraining. As reported in Table 1, RayWAM obtains an average success rate of 97.25% using a 1.3B video diffusion backbone, which is comparable to Fast-WAM's 97.6% despite using a much larger 6B model. Compared with methods relying on additional embodied pretraining, RayWAM achieves similarly strong manipulation performance using only a pretrained video diffusion backbone, demonstrating that aligning robot actions with the native visual space allows effective reuse of the rich visual dynamics priors learned by video diffusion models.

Generalization on LIBERO-Plus. We evaluate LIBERO-Plus in a zero-shot setting using the LIBERO-trained checkpoint without adaptation. As shown in Table 1, RayWAM achieves an average success rate of **67.84**% across seven distribution shifts, improving over Fast-WAM by 16.34 percentage points. It achieves the highest performance under the Language shift (**90.11**%) and strong performance under spatial shifts, including the Camera shift (**70.36**%) and Layout shift (**83.08**%). We attribute these gains to the combination of Action Raymap and joint world-action modeling. By representing executable actions in the native visual space of video diffusion models, Action Raymap enables better reuse of language-conditioned and visual dynamics priors for action generation. Meanwhile, jointly modeling future observations and actions allows the policy to leverage predicted visual dynamics when adapting to different viewpoints and spatial arrangements. The improvement is not uniform across all perturbations. Under the Background shift, RayWAM obtains 48.61% compared with 53.7% for Fast-WAM, indicating that task-irrelevant static background changes remain challenging.

Progress-Aware Manipulation on VLABench. We evaluate RayWAM on VLABench Primitive-10 FT using Success Rate (SR), Intention Score (IS), and Progress Score (PS), which jointly measure task completion, intention consistency, and execution progress. As shown in Table 2, RayWAM achieves 60.8% SR, 87.17% IS, and 74.75% PS. Compared with Fast-WAM, this corresponds to gains of 2.4, 5.97, and 0.25 percentage points, respectively, with RayWAM achieving the highest IS among the evaluated methods. These results indicate that joint world-action generation supports not only task completion but also more consistent task-directed behavior and progress, without introducing an explicit progress estimator.

Table 2: **Results on VLABench Primitive-10 FT.** Success Rate (SR), Intention Score (IS), and Progress Score (PS) are reported, which measure task completion, intention consistency, and execution progress, respectively. The best and second-best results are highlighted in **bold** and underlined.

Method	Type	Params	Emb. PT.	SR ↑	IS ↑	PS ↑
π_0	VLA	3.3B	Yes	47.0	67.8	62.7
π_0-FAST	VLA	3.3B	Yes	56.2	72.4	66.8
$\pi_{0.5}$	VLA	3.3B	Yes	**65.4**	80.4	**77.8**
Motus	WAM	8B	Yes	54.4	78.6	70.2
Fast-WAM	WAM	6B	No	58.4	81.2	74.5
Ours	WAM	**1.3B**	No	60.8	**87.17**	74.75

4.2 Real-world Experiments

We further evaluate RayWAM on four real-world manipulation tasks using a Franka Research 3 robot: *Place Fruit into Box*, *Insert Pen into Holder*, *Wipe Whiteboard*, and *Stack Three Bowls*. These tasks span pick-and-place, precision insertion, surface interaction, and long-horizon manipulation. We collect 100 demonstrations per task and train a single multi-task checkpoint across all four tasks. For comparison, π_0 and Fast-WAM are trained with the same demonstrations under the same multi-task training setting and evaluation protocol, ensuring a controlled comparison; full experimental details are provided in Appendix C.

Table 3: **Real-world manipulation and generalization results.** Success rates (%) are reported for four in-distribution tasks and the Layout OOD setting. Each task is evaluated over 30 independent trials with the same initial reset procedure.

				In-Distribution					Out-of-Distribution
Method	Type	Params	Emb. PT.	Fruit-to-Box	Pen Insert	Whiteboard Wipe	Bowl Stack	Avg.	Layout OOD
π_0	VLA	3.3B	Yes	86.7	70.0	46.7	70.0	68.3	46.7
Fast-WAM	WAM	6B	No	**93.3**	63.3	56.7	**93.3**	76.7	33.3
Ours	WAM	1.3B	No	**93.3**	**76.7**	**66.7**	83.3	**80.0**	**56.7**

Manipulation Performance. Table 3 reports performance on the four in-distribution real-world tasks. RayWAM achieves the highest average success rate among the evaluated methods. It achieves higher success rates on *Pen Insertion* and *Whiteboard Wiping* while matching Fast-WAM on Fruit-to-Box, although Fast-WAM performs better on Bowl Stacking. Overall, RayWAM demonstrates effective multi-task real-world manipulation using a single 1.3B policy.

Generalization. We further evaluate spatial generalization on *Place Fruit into Box* under the Layout OOD setting, where additional distractor objects are introduced and their spatial positions are randomized while preserving the task objective. As shown in Table 3, RayWAM achieves the highest Layout OOD success rate among the evaluated methods, indicating stronger robustness to changes in object arrangement and visual clutter.

4.3 Ablation Study

We conduct all ablations on LIBERO-Goal under the same training configuration, data budget, and Wan2.1-1.3B backbone, varying only the component under study. The image-space action-representation and attention-mask ablations use the original pretrained Wan2.1 VAE, whereas Latent-Repeat bypasses the action-side VAE by design. The VAE-adaptation ablation compares the raw and adapted decoders under the same policy setting. We additionally perform a ground-truth *VAE round-trip reconstruction* test ($\mathcal{X} \rightarrow \mathcal{M} \xrightarrow{\text{VAE}} \hat{\mathcal{M}} \rightarrow \hat{\mathcal{X}}$) for the image-space action-representation and VAE-adaptation variants (Appendix D.2) to quantify information loss introduced by the visual bottleneck independently of policy training. Results are summarized in Table 4.

Table 4: **Ablation results on LIBERO-Goal.** Image-space representation and attention-mask variants use the raw pretrained VAE; Latent-Repeat bypasses the action-side VAE. We report policy success rate (SR, %) and translation error (mm); complete reconstruction metrics are provided in Appendix D.2.

	Action Representation				**Attention Mask**			**VAE Adaptation**	
	Raymap	Lat.-Rep.	Flat.-Res.	Shuffled	Dir.	Decoup.	Bidir.	Raw	Adapt.
SR ↑	93.00	87.80	0.00	0.00	93.00	78.00	90.00	93.00	**98.20**
Trans. Err. ↓	4.57	–	70.76	60.32	–	–	–	4.57	**0.81**

Impact of Action Representation. We compare Action Raymap with three controlled alternatives. *Latent-Repeat* follows Cosmos Policy (Kim et al., 2026) by repeating normalized 8-D state/action vectors directly in latent space, bypassing the action-side VAE. *Flatten-Resize* repeats and reshapes the same vectors into CHW image tensors before VAE encoding, while *Shuffled Action Raymap* applies a fixed spatial permutation to the original Action Raymap, preserving its pixel values and channels while disrupting the layout. Detailed constructions are provided in Appendix D.1. As shown in Table 4, Flatten-Resize and Shuffled Action Raymap incur large VAE round-trip translation errors and both achieve 0.00% policy success, showing that RGB formatting alone is insufficient and that preserving the spatial structure of Action Raymap is important for VAE-compatible action encoding. *Latent-Repeat* avoids the VAE reconstruction bottleneck but achieves 87.80% success, compared with 93.00% for Action Raymap with the unadapted VAE, indicating that an image-native action representation better leverages the pretrained video model than direct latent repetition.

Impact of Attention Mask. We compare our directional attention with two alternatives: *Decoupled Mask*, which blocks interaction between future RGB and Action Raymap latents, and *Bidirectional Mask*, which allows attention in both directions. Decoupling reduces the success rate to 78.00%, showing that action generation benefits substantially from access to predicted visual dynamics. Bidirectional interaction improves performance to 90.00% but remains below the 93.00% achieved by our directional mask under the same Raw-VAE setting, indicating that allowing future visual prediction to attend to future actions is less effective than preserving a directional flow from visual dynamics to action generation. Detailed constructions are provided in Appendix D.3.

Effectiveness of VAE Adaptation. We evaluate the optional decoder-side VAE adaptation on LIBERO-Goal by comparing the original Wan2.1 VAE (*Raw VAE*) with our adapted variant, which fine-tunes the VAE decoder while keeping the encoder fixed. As shown in Table 4, the original VAE already achieves a success rate of 93.00%, indicating that Action Raymap is compatible with the pretrained video latent space. Decoder-side adaptation further improves the success rate to 98.20% and reduces the translation reconstruction error from 4.57 mm to 0.81 mm. Detailed quantitative comparisons of translation, rotation, and gripper reconstruction errors are provided in Appendix D.2.

5 Conclusion and Discussion

We introduced RayWAM, a World-Action Model that bridges pretrained video models and robot control at the representation level. Action Raymap reparameterizes executable end-effector actions as structured RGB maps, avoiding dependence on real-camera parameters or learned action decoding while preserving deterministic recovery. This enables observations and actions to share the same pretrained video VAE and diffusion backbone for joint world-action generation with minimal action-specific machinery. Across LIBERO, LIBERO-Plus, VLABench, and real-world tasks, RayWAM demonstrates effective manipulation and generalization using a compact 1.3B video backbone without additional embodied pretraining. These results suggest that adapting action representations to the native visual space of video models provides a simple path for transferring pretrained video priors to robot control. However, RayWAM has not yet undergone large-scale pretraining on diverse robot interaction data, and its inference speed remains relatively slow. Future work will explore large-scale robot interaction pretraining for broader generalization and faster inference methods for more responsive control.

AI USE STATEMENT

Generative AI tools were used only for language editing and stylistic refinement of the manuscript. All AI-assisted text was reviewed and revised by the authors for accuracy and clarity. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide detailed information for reproducing RayWAM in the main paper and appendices. Sec. 3 specifies the Action Raymap formulation and world-action generation framework. Appendix A contains implementation details, preprocessing, training and inference procedures, VAE adaptation, and hyperparameters; Appendix B documents the simulation evaluation protocols; Appendix C describes the real-world setup and evaluation; and Appendix D provides detailed ablation configurations. We will release the code and implementation details upon acceptance.

REFERENCES

Hadi Alzayer, Wenlong Huang, Haonan Chen, Christopher Luey, Lvmin Zhang, Maneesh Agrawala, Gordon Wetzstein, Li Fei-Fei, Yilun Du, Jiajun Wu, et al. Masked visual actions for unified world modeling. *arXiv preprint arXiv:2607.19343*, 2026.

Sherwin Bahmani, Ivan Skorokhodov, Aliaksandr Siarohin, Willi Menapace, Guocheng Qian, Michael Vasilkovsky, Hsin-Ying Lee, Chaoyang Wang, Jiaxu Zou, Andrea Tagliasacchi, et al. Vd3d: Taming large video diffusion transformers for 3d camera control. In *International Conference on Learning Representations*, volume 2025, pp. 66712–66737, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Yixiang Chen, Peiyan Li, Yuan Xu, Qisen Ma, Jiabing Yang, Kai Wang, Jianhua Yang, Dong An, He Guan, Gaoteng Liu, et al. Flowwam: Optical flow as a unified action representation for world action models. *arXiv preprint arXiv:2607.13017*, 2026a.

Yixiang Chen, Peiyan Li, Jiabing Yang, Keji He, Xiangnan Wu, Yuan Xu, Kai Wang, Jing Liu, Nianfeng Liu, Yan Huang, et al. Bridgev2w: Bridging video generation models to embodied world models via embodiment masks. *arXiv preprint arXiv:2602.03793*, 2026b.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

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

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for video diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 100433–100464, 2025a.

Hao He, Ceyuan Yang, Shanchuan Lin, Yinghao Xu, Meng Wei, Liangke Gui, Qi Zhao, Gordon Wetzstein, Lu Jiang, and Hongsheng Li. Cameractrl ii: Dynamic scene exploration via camera-controlled video diffusion models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13416–13426. IEEE, 2025b.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Wonbong Jang, Shikun Liu, Soubhik Sanyal, Juan Camilo Perez, Kam Woh Ng, Sanskar Agrawal, Juan-Manuel Perez-Rua, Yiannis Douratsos, and Tao Xiang. Rays as pixels: Learning a joint distribution of video and camera trajectories. In *Forty-third International Conference on Machine Learning*, 2026.

Yuxin Jiang, Shengcong Chen, Siyuan Huang, Liliang Chen, Pengfei Zhou, Yue Liao, Xindong He, Chiming Liu, Hongsheng Li, Maoqing Yao, et al. Enerverse-ac: Envisioning embodied environments with action condition. *arXiv preprint arXiv:2505.09723*, 2025.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

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

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

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.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Wei Liu, Ziyu Chen, Zizhang Li, Yue Wang, Hong-Xing Yu, and Jiajun Wu. Realwonder: Real-time physical action-conditioned video generation. *arXiv preprint arXiv:2603.05449*, 2026.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Sandeep Kumar Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions. In *International Conference on Learning Representations*, volume 2026, pp. 68823–68857, 2026.

Zhenhao Shen, Jiaqi Liang, Jasper Lu, Feng Jiang, Yuran Wang, Chuanbo Wei, Jiayi Liu, Jianchun Yang, Qize Yu, Jiadi You, et al. Ld4wam: Learning latent dynamics from human videos for world action models. *arXiv preprint arXiv:2608.22403*, 2026.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Junke Wang, Qihang Zhang, Shuai Yang, Yiming Luo, Yujun Shen, Zuxuan Wu, Yu-Gang Jiang, and Yinghao Xu. Repwam: World action modeling with representation visual-action tokenizers. *arXiv preprint arXiv:2606.13674*, 2026a.

Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, et al. Video generation models are general-purpose vision learners. In *European Conference on Computer Vision*, pp. 608–627. Springer, 2026b.

Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12713–12724. IEEE, 2025.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026c.

Xionghao Wu, Yijun Yang, Shiyang Zhou, Haoze Sun, Jianhui Liu, Songsong Yu, Jiyao Zhang, Wenbo Li, Bo Wang, Guoqing Ma, et al. Zimablue: Evolving generalizable world action models through scalable video pre-training. *arXiv preprint arXiv:2609.00188*, 2026.

Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vahdat. Camco: Camera-controllable 3d-consistent image-to-video generation. *arXiv preprint arXiv:2406.02509*, 2024.

Haodong Yan, Zhide Zhong, Jiaguan Zhu, Junjie He, Weilin Yuan, Wenxuan Song, Xin Gong, Yingjie Cai, Guanyi Zhao, Xu Yan, et al. S-vam: Shortcut video-action model by self-distilling geometric and semantic foresight. *arXiv preprint arXiv:2603.16195*, 2026.

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.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. Dreamwam: Beyond rgb future prediction for world action models. *arXiv preprint arXiv:2608.04996*, 2026a.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

Jason Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *International conference on learning representations*, volume 2024, pp. 23345–23366, 2024.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025.

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.

Guangcong Zheng, Teng Li, Rui Jiang, Yehao Lu, Tao Wu, and Xi Li. Cami2v: Camera-controlled image-to-video diffusion model. *arXiv preprint arXiv:2410.15957*, 2024.

APPENDIX

A IMPLEMENTATION DETAILS

This section provides additional implementation details for RayWAM. We describe the model architecture (Sec. A.1), the dataset-specific multi-view input composition (Sec. A.2), policy training (Sec. A.3), closed-loop inference (Sec. A.4), action recovery (Sec. A.5), Action Raymap-aware VAE adaptation (Sec. A.6), and the Action Raymap parameterization and hyperparameters (Sec. A.7).

A.1 MODEL ARCHITECTURAL COMPONENTS

Video Diffusion Backbone. RayWAM is initialized from the pretrained Wan2.1-T2V-1.3B model (Wan et al., 2025). We retain its native video DiT architecture, consisting of 30 transformer blocks with a hidden dimension of 1536, an FFN dimension of 8960, 12 attention heads, and a head dimension of 128. The DiT operates on the native 16-channel Wan2.1 latent space with a spatiotemporal patch size of $1 \times 2 \times 2$. We fully fine-tune the video DiT during policy training and introduce no action-specific prediction head, tokenizer, or transformer module.

Video VAE. We use the pretrained Wan2.1 video VAE as the shared visual tokenizer for both RGB observations and Action Raymaps. The same VAE encoder maps the two modalities into the native 16-channel latent space of the video DiT. For a 17-frame input sequence, the VAE produces five latent frames due to its temporal compression. The encoder remains frozen throughout policy training, preserving the pretrained latent representation for both modalities.

Language Encoder. We use the UMT5-XXL text encoder from the pretrained Wan2.1 model for language conditioning. Language instructions are tokenized with a maximum context length of 128 tokens and encoded into text features that condition the video DiT. The text encoder remains frozen throughout policy training.

A.2 DATASET-SPECIFIC MULTI-VIEW COMPOSITION

For each dataset, we concatenate all observation views horizontally to form a single RGB canvas. The corresponding Action Raymap tile is horizontally replicated to match the spatial resolution of the RGB input, ensuring identical three-channel spatiotemporal tensor shapes for the two modalities. The dataset-specific view composition and input resolutions are summarized in Table 5.

Table 5: **Dataset-specific multi-view composition.** Action Raymap tiles are repeated horizontally to match the spatial organization of the corresponding RGB canvas.

Dataset	Views	Per-view size	RGB canvas	Raymap copies
LIBERO / LIBERO-Plus	2	224×224	224×448	2
VLABench	3	192×192	192×576	3
Real-world	2	224×224	224×448	2

LIBERO and LIBERO-Plus. For LIBERO, we use the external observation view (`image`) and the wrist-mounted view (`wrist_image`), which are concatenated horizontally. The corresponding Action Raymap tile is duplicated horizontally to match the RGB canvas. LIBERO-Plus follows the same input composition as LIBERO, as it is evaluated zero-shot using the checkpoint trained on LIBERO.

VLABench. For VLABench, we use three observation streams: `image`, `second_image`, and `wrist_image`, which are concatenated horizontally. The corresponding Action Raymap tile is repeated three times along the horizontal dimension.

Real-world. For real-world experiments, we use a left workspace camera and a wrist-mounted camera. The two views are concatenated horizontally, and the Action Raymap tile is duplicated to match the resulting RGB canvas.

A.3 Training Details

Policy Optimization. During policy training, we fully fine-tune the pretrained video DiT while keeping the video VAE and UMT5-XXL text encoder frozen. We use AdamW with a cosine learning-rate schedule and linear warmup, and train with BF16 mixed precision and gradient clipping. We train for 10 epochs on LIBERO and the real-world datasets, and for 5 epochs on VLABench. Unless otherwise specified, we use an effective batch size of 128. Policy training is performed on 8 NVIDIA A800 GPUs. The complete optimization hyperparameters are summarized in Sec. A.7.

Language Prompt. For language conditioning, each task instruction is inserted into the fixed prompt template `A video recorded from a robot's point of view executing the following instruction: {task_instruction}`, where `task_instruction` is replaced by the natural-language instruction of the current task. The resulting prompt is tokenized and encoded by the frozen UMT5-XXL text encoder to obtain the language-conditioning features used by the video DiT.

Cached-Latent Training. Since the VAE encoder and UMT5-XXL text encoder remain fixed during policy training, we precompute the RGB latents, Action Raymap latents, and language embeddings before optimization. RGB and Action Raymap latents from the same temporal window are stored as paired BF16 tensors. During training, they are concatenated along the temporal dimension following the ordering $[\mathbf{I}_0, \mathbf{I}_{1:L-1}, \mathbf{S}_0, \mathbf{A}_{1:L-1}]$ defined in Sec. 3.3, and are directly provided to the video DiT together with the cached language features. This avoids repeated execution of the frozen encoders during policy optimization.

A.4 Inference Details

RayWAM is deployed in a receding-horizon closed loop. At each policy query, the current multi-view RGB observation is composed following Sec. A.2, and the current end-effector state is converted into the conditioning Action Raymap. The two modalities are independently encoded by the shared frozen VAE encoder to obtain the current RGB and state-conditioning latents.

Future RGB and Action Raymap latents are initialized from Gaussian noise and jointly denoised by the video DiT using the block-wise attention described in Sec. 3.3. Unless otherwise specified, we use 20 denoising steps for each policy query.

Since future RGB reconstruction is not required for control, we decode only the generated Action Raymap latents and deterministically recover the corresponding relative action chunk through Φ^{-1}. The replicated Action Raymap tiles are first averaged pixel-wise, after which translation, rotation, and gripper state are recovered from the resulting averaged map. The complete recovery procedure is described in Sec. A.5.

Each policy query predicts a 16-step action chunk. We execute the first 8 actions and then replan from the latest observation. This receding-horizon procedure is used throughout our evaluations unless otherwise specified.

A.5 Action Recovery Details

Aggregation of Replicated Action Raymaps. As described in Sec. A.2, each Action Raymap tile is horizontally replicated to match the spatial dimensions of the corresponding multi-view RGB canvas. After VAE decoding, we split the decoded Action Raymap canvas into its N replicated tiles, denoted by $\{\hat{\mathbf{m}}_{t+k}^{(j)}\}_{j=1}^{N}$, and average corresponding pixels across all copies:

$$\bar{\mathbf{m}}_{t+k} = \frac{1}{N} \sum_{j=1}^{N} \hat{\mathbf{m}}_{t+k}^{(j)}. \tag{11}$$

Numerical action recovery is then performed once from the averaged Action Raymap $\bar{\mathbf{m}}_{t+k}$.

The relative translation is recovered by taking the channel-wise median over the valid pixels in the central region $\mathcal{C}$ of $\bar{\mathbf{m}}_{t+k}$, followed by inverse rescaling using the relative-action normalization

statistics. The discrete gripper state is extracted from the boundary region $\mathcal{B}$. For rotation recovery, the predicted three-dimensional vectors in the directional region $\mathcal{D}$ are individually normalized to unit length before fitting a single rotation matrix to the canonical ray field.

Closed-Form Rotation Recovery via SVD. Let $\tilde{\mathbf{d}}_{t,\mathbf{u}}^{(k)} \in \mathbb{R}^3$ denote the predicted directional vector at pixel $\mathbf{u} \in \mathcal{D}$ extracted from the averaged Action Raymap $\bar{\mathbf{m}}_{t+k}$. Each predicted vector is first normalized to unit length, yielding $\hat{\mathbf{d}}_{t,\mathbf{u}}^{(k)}$. Given the corresponding canonical unit direction $\mathbf{d}_\mathbf{u}$, we recover the relative rotation by solving

$$\widehat{\mathbf{R}}_t^{(k)} = \arg\min_{\mathbf{R}\in \mathrm{SO}(3)} \sum_{\mathbf{u}\in\mathcal{D}} \left\| \hat{\mathbf{d}}_{t,\mathbf{u}}^{(k)} - \mathbf{R}\mathbf{d}_\mathbf{u} \right\|_2^2. \tag{12}$$

This rotation-fitting problem admits a closed-form solution. We first form the 3×3 correlation matrix

$$\mathbf{H}_t^{(k)} = \sum_{\mathbf{u}\in\mathcal{D}} \hat{\mathbf{d}}_{t,\mathbf{u}}^{(k)} \mathbf{d}_\mathbf{u}^\top, \tag{13}$$

and compute its singular value decomposition

$$\mathbf{H}_t^{(k)} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{14}$$

where $\mathbf{U}$ and $\mathbf{V}$ contain the left and right singular vectors, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix of singular values. Defining

$$\mathbf{S} = \mathrm{diag}\left(1,\, 1,\, \det(\mathbf{U}\mathbf{V}^\top)\right), \tag{15}$$

the recovered rotation is

$$\widehat{\mathbf{R}}_t^{(k)} = \mathbf{U}\mathbf{S}\mathbf{V}^\top. \tag{16}$$

The determinant correction in $\mathbf{S}$ ensures that $\widehat{\mathbf{R}}_t^{(k)} \in \mathrm{SO}(3)$ and avoids a reflected solution. Finally, $\widehat{\mathbf{R}}_t^{(k)}$ is converted to a unit quaternion and combined with the recovered translation and gripper state to obtain the relative action.

A.6 Action Raymap-Aware VAE Adaptation

We optionally adapt the decoder side of the pretrained Wan2.1 VAE to improve Action Raymap reconstruction. This adaptation is implemented as a standalone VAE reconstruction procedure and is fully independent of video DiT training: only the pretrained Wan2.1 VAE is loaded, while the video DiT is not involved. Since the VAE encoder remains fixed, the adaptation does not alter the latent representation used for policy learning and imposes no ordering dependency on DiT optimization.

Trainable Components. We keep the VAE encoder frozen, and optimize only the VAE decoder. Denoting the trainable decoder-side parameters by ψ, the reconstruction process is

$$\mathcal{Z}^m = \mathcal{E}(\mathcal{M}), \qquad \hat{\mathcal{M}} = \mathcal{G}_\psi(\mathcal{Z}^m), \tag{17}$$

where $\mathcal{E}$ denotes the frozen pretrained VAE encoder and $\mathcal{G}_\psi$ denotes the trainable decoder-side mapping.

We optimize the Action Raymap reconstruction objective

$$\mathcal{L}_{\mathrm{VAE}} = \sum_{\mathcal{R}\in\{\mathcal{C},\mathcal{D},\mathcal{B}\}} \left\| \hat{\mathcal{M}} - \mathcal{M} \right\|_{1,\mathcal{R}}, \tag{18}$$

where $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{B}$ denote the translation, ray-direction, and gripper regions, respectively, and $\|\cdot\|_{1,\mathcal{R}}$ denotes the mean absolute reconstruction error over region $\mathcal{R}$. No auxiliary loss is applied to the recovered low-dimensional actions.

Optimization. VAE adaptation reuses the cached Action Raymap latents prepared with the same frozen VAE encoder used for policy training, following Sec. A.3. The cached latents are directly provided to the trainable decoder-side components, making VAE adaptation independent of both the video DiT and its optimization schedule. When adaptation is enabled, the adapted decoder-side components are used for Action Raymap reconstruction, while the encoder and the policy latent representation remain unchanged. Detailed optimization settings are summarized in Sec. A.7.

A.7 Action Raymap Parameterization and Hyperparameters

Action Raymap Parameterization. For an $h \times w$ Action Raymap tile, we use a fixed virtual-camera intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} -\frac{fw}{2} & 0 & \frac{w-1}{2} \\ 0 & -\frac{fh}{2} & \frac{h-1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \qquad f = 0.2, \tag{19}$$

where the principal point coincides with the geometric center of the virtual image plane. The canonical direction associated with pixel $\mathbf{u} = (u, v)$ is obtained by inverse projection followed by unit normalization:

$$\mathbf{d}_{\mathbf{u}} = \frac{\mathbf{K}^{-1}[u, v, 1]^\top}{\|\mathbf{K}^{-1}[u, v, 1]^\top\|_2}. \tag{20}$$

The resulting canonical ray field depends only on the Action Raymap tile resolution and the fixed focal parameter f, and is independent of the physical camera calibration, scene, or dataset statistics.

The central translation region $\mathcal{C}$ occupies 50% of the tile width and 50% of its height, corresponding to 25% of the tile area. All pixels within this region encode the same three-dimensional translation vector.

Hyperparameter Summary. Table 6 summarizes the default architectural, optimization, Action Raymap, and inference settings used in our main experiments unless otherwise specified.

Table 6: **Hyperparameter summary for RayWAM.**

Category	Hyperparameter	Value
Model / Raymap	Video backbone	Wan2.1-T2V-1.3B
	Action horizon H	16
	Virtual-camera focal parameter f	0.2
	Translation normalization range	$[-1, 1]$
	Central-region width/height fraction	0.5
Training	Optimizer	AdamW
	Learning rate	1×10^{-4}
	AdamW (β_1, β_2)	$(0.9, 0.95)$
	Weight decay	1×10^{-2}
	Maximum gradient norm	1.0
	Learning-rate schedule	Cosine
	Warmup ratio	5%
	Training epochs	10 (default); 5 (VLABench)
	Effective batch size	128
	Timestep shift	5.0
	λ_{rgb}	1.0
	λ_{Action}	5.0
Inference	Timestep shift	5.0
	Denoising steps	20
	Actions executed before replanning	8
VAE adapt.	Optimizer	AdamW
	Learning rate	1×10^{-5}
	Weight decay	0
	Learning-rate schedule	Cosine
	Warmup ratio	5%
	Training epochs	2
	Effective batch size	64

B Additional Simulation Evaluation Details and Results

This section provides supplementary evaluation details for LIBERO, LIBERO-Plus, and VLABench, including benchmark-specific protocols, inference settings, metric definitions, and detailed quantitative results.

B.1 LIBERO

Evaluation Protocol. We train a single RayWAM checkpoint jointly on demonstrations from the four standard LIBERO suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. The same checkpoint is evaluated across all four suites without suite-specific fine-tuning. Each suite contains 10 tasks, and we conduct 50 evaluation rollouts per task, resulting in 500 rollouts per suite and 2,000 rollouts in total. A rollout is considered successful when the benchmark-provided task completion condition is satisfied.

We follow the closed-loop inference procedure in Appendix A.4. For each suite, we report the average success rate over its 10 tasks, and the overall LIBERO score is computed as the average of the four suite-level success rates.

Per-Suite Results. Table 7 reports the complete per-suite results corresponding to the LIBERO evaluation in the main text.

Table 7: **General manipulation performance on LIBERO.** We report success rates (%) on four task suites and their average. Params denotes model size in billions of parameters; for our method, we report the size of the video DiT backbone. Emb. PT. denotes pretraining on additional embodied robot data beyond the benchmark demonstrations. Higher success rates are better. Bold and underlined values indicate the best and second-best results in each success-rate column, including ties. – denotes an unavailable result.

Method	Params	Emb. PT.	Spatial	Object	Goal	Long	Avg.
Vision-Language-Action Models							
UniVLA	7B	Yes	96.5	96.8	95.6	92.0	95.2
OpenVLA-OFT	7B	Yes	97.6	98.4	97.9	94.5	97.1
π_0	3.3B	Yes	96.8	98.8	95.8	85.2	94.2
π_0-FAST	3.3B	Yes	96.4	96.8	88.6	60.2	85.5
$\pi_{0.5}$	3.3B	Yes	<u>98.8</u>	98.2	<u>98.0</u>	92.4	96.9
World-Action Models							
VLA-JEPA	2B	Yes	96.2	<u>99.6</u>	97.2	<u>95.8</u>	97.2
w/o human videos	2B	Yes	94.8	<u>99.6</u>	95.8	94.0	96.1
Fast-WAM	6B	No	98.2	**100.0**	97.0	95.2	<u>97.6</u>
OpenWAM-α	6B	Yes	**99.6**	<u>99.6</u>	**99.8**	**98.2**	**99.3**
Ours	**1.3B**	No	97.8	98.2	<u>98.0</u>	95.0	97.25

B.2 LIBERO-PLUS

Zero-Shot Evaluation Protocol. We evaluate RayWAM on LIBERO-Plus in a zero-shot setting using the checkpoint trained on the original LIBERO benchmark. The same observation preprocessing, Action Raymap construction, and normalization statistics used for LIBERO are retained for evaluation.

LIBERO-Plus applies seven types of distribution shifts to tasks from the four original LIBERO suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Following the benchmark protocol, each perturbed task instance is evaluated with a single rollout, resulting in 10,030 evaluation episodes in total. We use the closed-loop inference procedure described in Appendix A.4. Success is determined by the benchmark-provided task completion criterion. We report success rates for each distribution shift together with the aggregate success rate over all evaluated instances.

Distribution Shifts. The seven perturbation dimensions are:

- **Camera:** changes in the observation viewpoint;
- **Robot:** variations in the robot configuration;
- **Language:** alternative natural-language descriptions of the same task;
- **Light:** changes in scene illumination;

- **Background:** changes in background appearance;
- **Noise:** visual corruption applied to the observations;
- **Layout:** changes in the spatial arrangement of objects and scene elements.

The corresponding quantitative results are reported in Table 1.

B.3 VLABench

Evaluation Protocol. We evaluate RayWAM on VLABench Primitive-10 FT using the official Track-1 in-distribution protocol. A single RayWAM checkpoint is jointly trained on demonstrations from all ten primitive tasks and evaluated on the same task set. We use the closed-loop inference procedure described in Appendix A.4.

Evaluation Metrics. Following the official VLABench evaluation, we report Success Rate (SR), Intention Score (IS), and Progress Score (PS). SR measures task completion, IS measures consistency with the intended task objective, and PS measures progress toward task completion. IS and PS therefore provide additional information for partially completed trajectories beyond binary success. All three metrics are computed using the benchmark-provided evaluator.

The aggregate results on Primitive-10 FT are reported in Table 2.

C Real-World Experimental Setup

This section provides additional details of the real-world experiments in Sec. 4.2, including the robot platform and sensing system, the evaluated tasks, their success criteria, and the evaluation protocol.

C.1 Robot Platform and Sensors

All real-world experiments are conducted using a Franka 3 robotic arm equipped with a Robotiq 2F-85 gripper. We use two Intel RealSense D435i cameras, consisting of a left workspace camera and a wrist-mounted camera. All methods are evaluated on the same robot platform. Inference is performed on an NVIDIA A800 GPU.

C.2 Real-World Tasks

We evaluate four real-world manipulation tasks: **Place Fruit into Box**, **Insert Pen into Holder**, **Wipe Whiteboard**, and **Stack Three Bowls**. These tasks cover object transport, precision insertion, sustained surface interaction, and sequential long-horizon manipulation, respectively. For each task, we collect 100 demonstrations, resulting in 400 demonstrations in total. A single multi-task checkpoint is jointly trained on the combined dataset.

Figure 4 shows representative demonstration sequences for the four real-world tasks. The corresponding scene-level task illustrations are provided in the main text.

C.3 Task Definitions and Success Criteria

We describe the task definitions, language instructions, and success criteria for all real-world tasks below. Success or failure is manually determined at the end of each evaluation trial according to the corresponding task criterion.

Place Fruit into Box. The robot is required to identify the target fruit specified by the language instruction, grasp it, transport it to the target box, and release it inside the box. In the in-distribution setting, we consider two target fruits, a mango and a bitter melon, with the corresponding language instructions:

- *"Pick up the mango and place it into the box."*
- *"Pick up the bitter melon and place it into the box."*

A trial is considered successful when the specified target fruit is completely contained inside the box at the end of the execution.

For the Layout OOD setting, the target fruit and its language instruction remain unchanged. We instead introduce additional distractor objects and randomize their spatial arrangement in the workspace. The success criterion remains identical to the in-distribution setting: the fruit specified by the instruction must be completely placed inside the target box.

Insert Pen into Holder. The robot is required to grasp the pen, move it toward the pen holder, align it with the holder opening, and insert it into the holder. We use the language instruction:

- *"Pick up the pen and insert it into the holder."*

A trial is considered successful when the pen is inserted into and remains inside the holder at the end of the execution.

Wipe Whiteboard. The robot is required to manipulate the wiping tool and remove the drawing from the whiteboard through continuous surface interaction. We use the language instruction:

- *"Pick up the eraser and wipe the drawing off the whiteboard."*

A trial is considered successful when the target drawing has been erased from the whiteboard at the end of the execution.

Stack Three Bowls. The robot is required to sequentially manipulate three bowls and place them on top of one another to form a stable stack. We use the language instruction:

- *"Pick up the bowls one by one and stack all three bowls."*

A trial is considered successful when all three bowls are stably stacked together at the end of the execution.

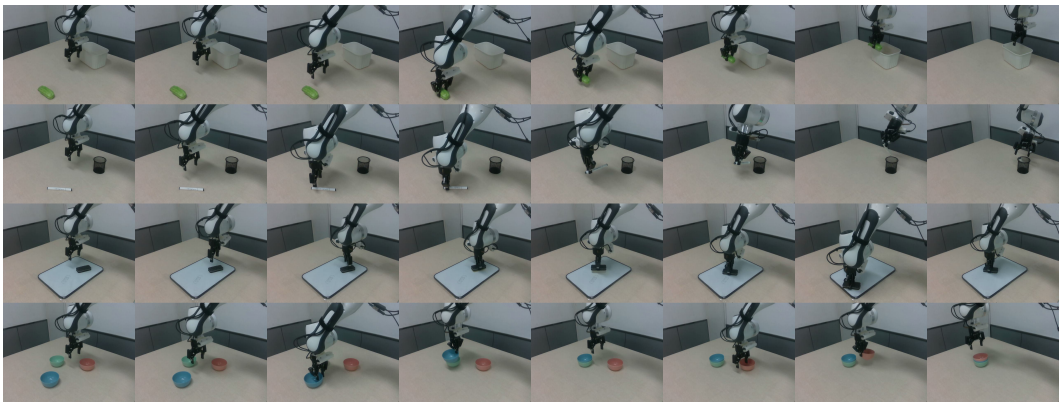

Figure 4: **Representative demonstration sequences for the four real-world tasks.** From top to bottom: **Place Fruit into Box**, **Insert Pen into Holder**, **Wipe Whiteboard**, and **Stack Three Bowls**. Each row shows representative frames from a demonstration trajectory, illustrating the manipulation process from the initial configuration to task completion.

C.4 Evaluation Protocol

For each task, we perform 30 independent evaluation trials. The robot and objects are reset according to the corresponding task-specific evaluation protocol before each trial. Success or failure is manually determined according to the task-specific criteria defined in Appendix C.3. All methods are evaluated using the same demonstration dataset and the same two-camera visual observations.

D Additional Ablation Studies

D.1 Action Representation Variants

We consider three alternative action representations to isolate the effect of the proposed Action Raymap design. Figure 5 visualizes the two image-space variants, Flatten-Resize and Shuffled Action Raymap, together with the original Action Raymap for reference. Their detailed constructions are described below.

Latent-Repeat. Cosmos Policy flattens a normalized action chunk, tiles it into a single latent frame in CHW order, and recovers actions by averaging repeated copies, bypassing the visual VAE for numerical inputs. We retain this tiling-and-averaging mechanism but adapt its temporal layout to our Wan2.1 baseline: one latent frame represents the observed state, while four future latent frames each encode four actions, yielding a 16-step horizon. Each state/action uses eight values—normalized translation, a unit quaternion, and gripper state—with absolute current-state poses and future poses relative to the current end effector. Each latent frame has shape $16 \times 28 \times 56$; the current-state vector is repeated 3,136 times, and each four-action vector is repeated 784 times. For a controlled representation ablation, we retain our baseline's RGB VAE, attention mask, diffusion objective, and RGB/action loss weights of $1{:}5$.

Flatten-Resize. Flatten-Resize preserves the same temporal sequence and pose conventions as Action Raymap: one current-state frame followed by 16 future-action frames. Each frame has its own 8-D vector $[x, y, z, q_w, q_x, q_y, q_z, g]$. The current-state pose is expressed in absolute coordinates, while each future target pose is expressed relative to the current end-effector pose. Translation is normalized to $[-1, 1]$ using the corresponding absolute-state or relative-action bounds; the unit quaternion uses a canonical sign, and the gripper value is mapped to $[-1, 1]$.

For each frame independently, its 8-D vector is repeated $18{,}816$ times and reshaped in CHW order into a $3 \times 224 \times 224$ tensor. This tile is duplicated horizontally to form a $3 \times 224 \times 448$ image. The 17 images retain their original temporal order and are jointly encoded by the original Wan2.1 VAE, exactly as in Action Raymap.

Recovery is also performed independently for each decoded frame. Both tiles are flattened in CHW order and regrouped into 8-D vectors. Corresponding components are averaged over repetitions and tiles, without averaging across time. Translation and gripper normalization are inverted, and the quaternion is re-normalized. Future poses are then transformed back to world coordinates using the observed current end-effector pose.

Shuffled Action Raymap. Shuffled Action Raymap preserves the same temporal sequence, pose conventions, and image dimensions as the standard Action Raymap. For each frame, we first construct the original $3 \times 224 \times 224$ Action Raymap tile and apply a fixed permutation over its 224×224 pixel coordinates, while moving the three channels of each pixel together. The same permutation is used across all frames and samples, with permutation seed 42. The shuffled tile is then duplicated horizontally to form a $3 \times 224 \times 448$ image, matching the input layout used for LIBERO.

The resulting 17-frame sequence is encoded and decoded by the original frozen Wan2.1 VAE. For recovery, the decoded image is split into two 224×224 tiles, and the exact inverse permutation is applied independently to each tile before the standard Action Raymap recovery procedure. The permutation is therefore fully reversible; any remaining round-trip error reflects distortion introduced by the pretrained VAE when the spatial structure of Action Raymap is disrupted.

D.2 Ground-Truth Round-Trip Reconstruction

Evaluation Protocol. We evaluate the reconstruction fidelity of the image-space action representations through a ground-truth VAE round trip on LIBERO-Goal. We use the held-out validation split constructed for VAE adaptation; these episodes are not used to optimize the adapted VAE. The validation split contains two demonstration episodes for each of the 10 tasks, yielding 20 episodes in total.

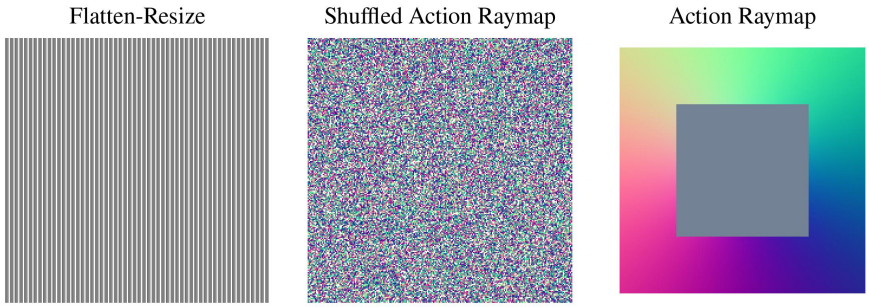

Figure 5: **Alternative action representations used in the representation ablation.** Flatten-Resize represents repeated numerical action vectors as RGB images, whereas Shuffled Action Raymap preserves the values of Action Raymap but disrupts their spatial organization using a fixed pixel permutation. The original Action Raymap is shown for reference.

For each validation episode, we consider every time step as a candidate starting point and retain only those for which a complete 17-frame sequence is available. Each sequence consists of the current end-effector state expressed in absolute coordinates, followed by 16 future action targets expressed relative to that current state. Translation is normalized using separate statistics for the absolute current state and relative future actions. Each ground-truth sequence is converted into its corresponding image-space representation, encoded and decoded by the VAE, and then mapped back to numerical actions using the representation-specific recovery procedure. The recovered actions are compared directly with the ground-truth actions to measure translation, rotation, and gripper reconstruction errors.

For the original-VAE setting, we use the pretrained Wan2.1 VAE without adaptation. For the adapted setting, we evaluate Action Raymap using the decoder-side adaptation described in Appendix A.6, with the same frozen encoder and held-out validation split.

Reconstruction Metrics. We compute reconstruction errors over all valid future action steps, excluding the current-state frame and padded steps. Translation error is defined as the Euclidean distance between the recovered and ground-truth end-effector target positions:

$$e_{\text{trans}} = \|\hat{\mathbf{p}} - \mathbf{p}\|_2 \times 1000, \tag{21}$$

where the error is reported in millimeters.

Rotation error is measured by the minimum relative rotation angle between the recovered and ground-truth orientations. After normalizing both quaternions, we compute

$$e_{\text{rot}} = 2\arccos\big(\text{clip}\left(\left|\hat{\mathbf{q}}^\top \mathbf{q}\right|, 0, 1\right)\big) \times \frac{180}{\pi}, \tag{22}$$

where the absolute inner product accounts for the equivalence between $\mathbf{q}$ and $-\mathbf{q}$. This metric therefore measures the relative orientation error directly rather than differences between individual Euler angles.

Gripper error is the classification error rate of the recovered binary open/close state, rather than the numerical reconstruction error of the gripper value. For translation and rotation, we report the mean and 95th-percentile (P95) errors over all valid future action steps; gripper error is reported as the percentage of incorrectly recovered gripper states.

Results. Table 8 reports the ground-truth round-trip reconstruction errors for the alternative image-space representations and Action Raymap under the original and adapted VAE settings.

D.3 Attention Variants

We compare two alternative cross-modal attention patterns against our default directional design. All variants use the same latent sequence $[\mathbf{I}_0, \mathbf{I}_{1:L-1}, \mathbf{S}_0, \mathbf{A}_{1:L-1}]$, Wan2.1-T2V-1.3B backbone,

Table 8: **Ground-truth action round-trip reconstruction on LIBERO-Goal.** We report mean and P95 translation and rotation errors, together with gripper error.

Representation	VAE	Trans. Err. (mm) ↓		Rot. Err. (°) ↓		Grip. Err. (%) ↓
		Mean	P95	Mean	P95	
Flatten-Resize	Original	70.764	113.751	122.595	171.700	12.583
Shuffled Action Raymap	Original	60.317	103.808	25.711	63.315	0.662
Action Raymap	Original	4.572	10.149	1.191	2.442	0.000
Action Raymap	Adapted	**0.807**	**1.679**	**0.113**	**0.257**	**0.000**

training data, flow-matching objective, and loss weights, and differ only in the attention connections between the future RGB latents $\mathbf{I}_{1:L-1}$ and future Action Raymap latents $\mathbf{A}_{1:L-1}$. Access to the conditioning latents $\mathbf{I}_0$ and $\mathbf{S}_0$ remains unchanged across all variants.

Decoupled Mask. The Decoupled Mask removes cross-modal attention between $\mathbf{I}_{1:L-1}$ and $\mathbf{A}_{1:L-1}$ in both directions. Future RGB and Action Raymap latents are therefore generated independently of each other while sharing the same current observation, robot-state, and language conditioning. This variant preserves the same latent organization and conditioning pathway as the default model, but prevents information exchange between the two future modalities during denoising. It therefore isolates the contribution of cross-modal interaction to joint world-action generation.

Bidirectional Mask. The Bidirectional Mask allows attention between $\mathbf{I}_{1:L-1}$ and $\mathbf{A}_{1:L-1}$ in both directions. Future RGB and Action Raymap latents can therefore exchange information throughout the denoising process, in contrast to the one-way interaction used by the default directional design. All other attention connections and conditioning inputs remain unchanged, so the only difference from the default model is that future RGB latents are additionally allowed to attend to future Action Raymap latents. This variant tests whether unrestricted interaction between the two future modalities is beneficial compared with preserving a directional dependency from visual dynamics to action generation.

Table 9: **Cross-modal attention variants.** Only the attention connections between future RGB latents $\mathbf{I}_{1:L-1}$ and future Action Raymap latents $\mathbf{A}_{1:L-1}$ are varied, while the latent sequence, conditioning inputs, and all other attention connections remain unchanged. The Decoupled Mask removes interaction between the two future modalities, the Bidirectional Mask enables interaction in both directions, and the Directional Mask retains only the dependency from future RGB latents to future Action Raymap latents.

Variant	$\mathbf{I}_{1:L-1}$ attends to $\mathbf{A}_{1:L-1}$	$\mathbf{A}_{1:L-1}$ attends to $\mathbf{I}_{1:L-1}$
Decoupled Mask	✗	✗
Bidirectional Mask	✓	✓
Directional Mask (Ours)	✗	✓

E Visualization Results

Figure 6 presents representative qualitative results of RayWAM across simulation and real-world evaluation. The visualization covers joint RGB–Action Raymap prediction on LIBERO, zero-shot generalization under a LIBERO-Plus background shift, task execution on VLABench, and real-world manipulation. For each example, predicted future RGB frames are shown together with the observations realized during closed-loop execution and the corresponding predicted Action Raymaps.

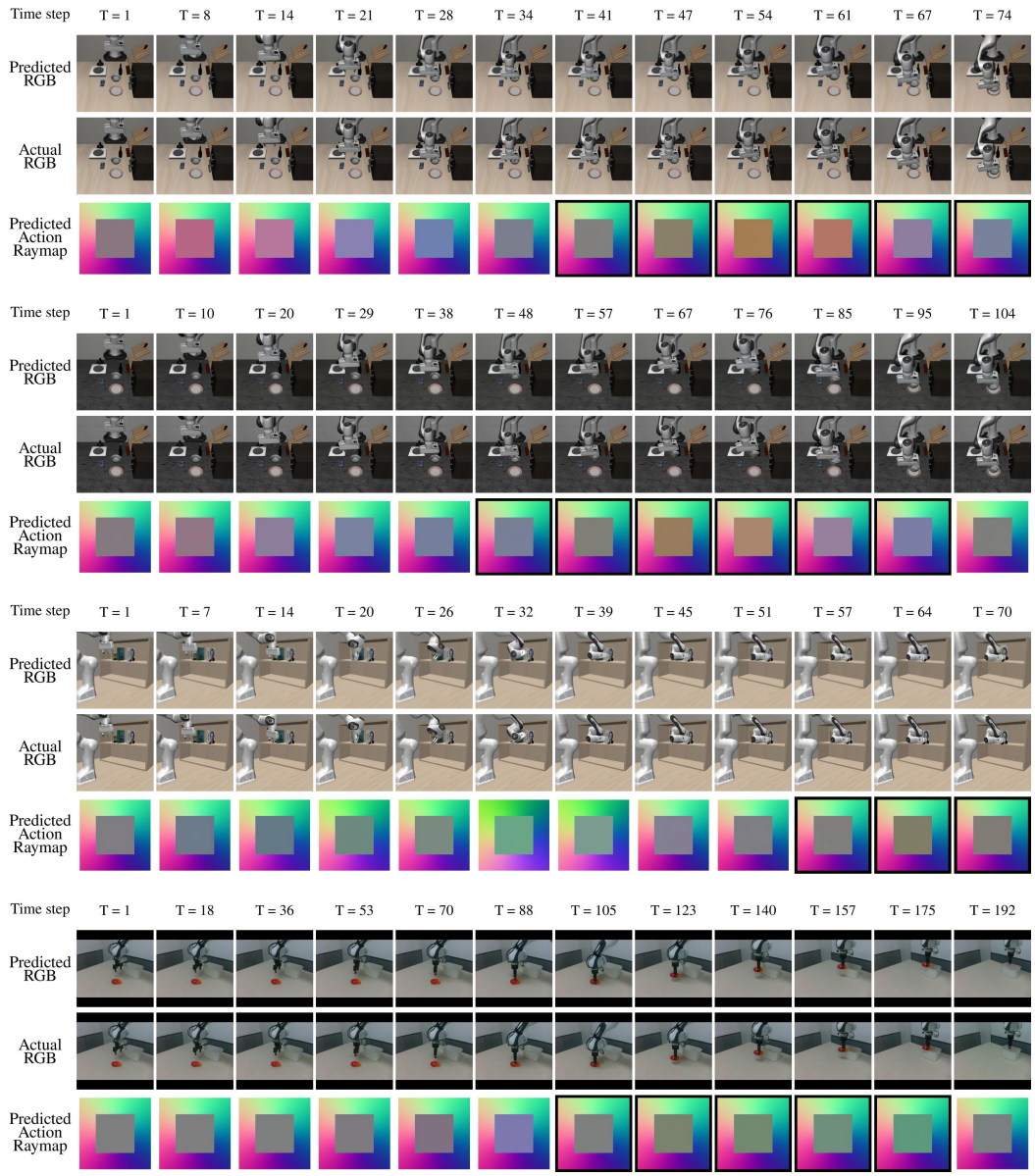

Figure 6: **Representative qualitative results of RayWAM.** From top to bottom, the four examples show: LIBERO-Goal with the instruction *"put the bowl on the plate"*; the same task under the LIBERO-Plus Background shift using the LIBERO-trained checkpoint; a VLABench Primitive-10 FT execution with the instruction *"Please take the book steve_jobs"*; and a real-world rollout in which the robot places the target mango into the box. Within each example, the three rows correspond to predicted future RGB frames, observations realized during closed-loop execution, and predicted Action Raymaps, respectively, with columns aligned by execution time. In the Action Raymaps, a white boundary indicates an open gripper and a black boundary indicates a closed gripper.