ACTIONROPE: ACTION AS COORDINATES FOR SPATIALLY CONSISTENT CONTROL IN PLANAR-CAMERA GAME WORLD MODELS

Anonymous authors
Paper under double-blind review

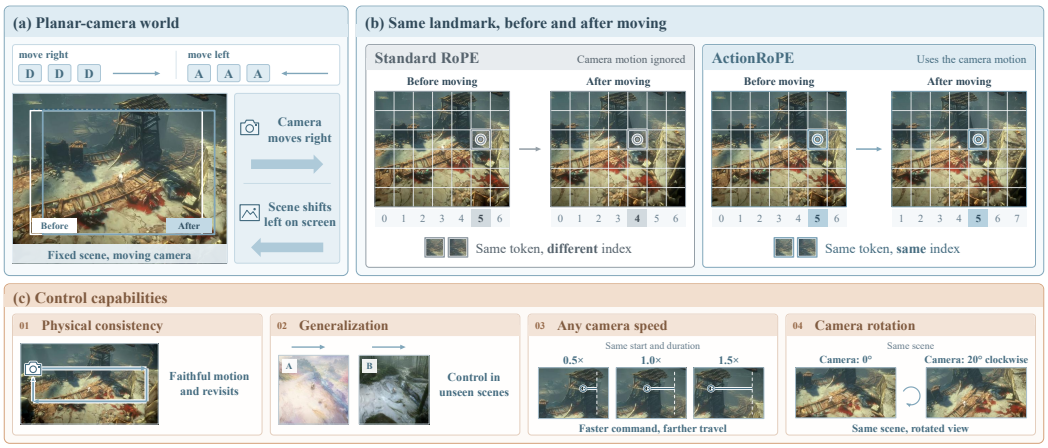

Figure 1: **Action as world coordinates.** (a) In a planar-camera world, each action corresponds to a known translation of the scene. (b) Standard RoPE re-addresses the same scene token after the camera moves, while ActionRoPE adds the camera displacement to each token's position, keeping its index across frames. (c) ActionRoPE enables precise motion and revisits, control in unseen scenes, and variable camera speed, without adding parameters to the control path.

ABSTRACT

Game world models have made rapid progress toward real-time interactive generation. Yet action control is still predominantly implemented as feature conditioning, leaving the model to learn from training data how each action should move the visual world. This formulation is poorly matched to a broad class of *planar-camera worlds*, such as top-down and isometric games. In these worlds, player motion induces a geometrically well-defined planar camera displacement. We observe that the key to spatially consistent control in such worlds is to represent the effect of each action directly in world coordinates. To this end, we introduce **ActionRoPE**, a parameter-free action interface that converts action-driven camera displacement into world-coordinate rotary positions. As a result, tokens corresponding to the same world location retain the same positional address across time. These world coordinates further expose a spatial asymmetry in generation: regions covered by the initial observation are geometrically known, while newly revealed regions remain to be synthesized. We therefore propose **Coordinate-Conditioned Noise Scheduling (CNS)**, which assigns token-wise denoising schedules according to this known–new structure. It allows observed regions to stabilize earlier, while new content is generated with reference to them. Compared with four representative action-conditioning mechanisms, our approach follows the commanded motion more faithfully and reduces the revisit drift by 66% over the strongest baseline. The same interface transfers across video backbones from 1.3B to 33B parameters and to out-of-distribution scenes. Project Page: https://anonymous.4open.science/w/ActionRoPE-7722/

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.

1 INTRODUCTION

Game world models are emerging as a promising paradigm for interactive visual simulation (Bruce et al., 2024; Parker-Holder et al., 2024). By conditioning video generation on user actions, they allow users to explore and manipulate generated worlds. Recent advances have substantially improved visual fidelity, temporal coherence, and generation efficiency, bringing these models closer to real-time playable environments (Valevski et al., 2025; Decart & Etched, 2024; Parker-Holder & Fruchter, 2025). Yet realistic video alone is not enough for interaction. The generated world must also respond to user actions accurately and in a spatially consistent way.

Existing game world models typically incorporate user actions as additional conditioning signals. Depending on the architecture, actions may be represented as discrete (Bruce et al., 2024) or textual (Zhu et al., 2026) tokens, or projected into the model as continuous features (Wang et al., 2026a). They may also modulate intermediate activations through adaptive normalization (Alaya-World Team et al., 2026) or enter through dedicated cross-attention modules (Wang et al., 2026b). Despite their different implementations, these approaches share the same underlying formulation: an action is encoded as a feature, and the generative model is expected to learn how that feature should control the visual world. In other words, both the direction and magnitude of the resulting motion are learned from data rather than explicitly specified by the control interface. This raises a fundamental question: *How should action control be represented when its visual effect is governed by explicit geometry instead of being learned from conditioning features?*

This question becomes particularly relevant in *planar-camera worlds*, such as top-down and isometric games. In these worlds, the player often stays near the screen center while the camera follows the player, causing the static scene to translate almost rigidly in the image plane. In this setting, the effect of an action is largely geometric. Once the action-to-camera scale is calibrated, the action directly determines both the direction and magnitude of the camera displacement. Yet existing interfaces still require the model to learn this geometric translation from data. As a result, the generated motion tends to regress toward motion patterns observed during training, failing to precisely follow the requested displacement. More fundamentally, feature-based action representations do not provide the model with an explicit world coordinate system. Without such coordinates, the model cannot directly tell whether a current token refers to a world location that has already been observed. When the camera returns to a previously seen location, the scene there may no longer look the same. The problem is therefore not merely insufficient conditioning capacity, but a mismatch between feature-based action representations and the geometric structure of planar-camera worlds.

To address this mismatch, we propose **ActionRoPE**, a parameter-free action interface for planar-camera world models. Instead of encoding an action as an additional feature, ActionRoPE writes the resulting camera displacement directly into the positional coordinates of video tokens. Concretely, it converts screen-space RoPE (Su et al., 2024) into action-aligned world-space RoPE, so that tokens corresponding to the same world location retain the same positional address across time (Figure 1). In this way, the action directly specifies the direction and magnitude of the resulting motion, enabling precise and geometric control. At the same time, the world coordinates naturally reveal which tokens belong to the scene already determined by the first frame and which belong to newly exposed regions. Building on this structure, we further introduce **Coordinate-Conditioned Noise Scheduling (CNS)**. Unlike applying an undifferentiated denoising schedule to all spatial tokens, CNS assigns token-wise schedules (Chen et al., 2024; Ruhe et al., 2024; Kim et al., 2024) according to a known–new partition. Specifically, as the camera moves, tokens inside the first-frame region are denoised and stabilized earlier, while newly revealed tokens keep the original schedule. This allows newly revealed regions to be generated with reference to the already settled scene. Besides a 0.015M-parameter input channel that feeds the known–new partition to the network, both components add no learnable parameters and leave the underlying attention architecture unchanged.

We evaluate ActionRoPE against four representative action-conditioning strategies on the same backbone and under the same training setting. Across motion scaling, directional compliance, and closed-loop revisit, our method yields more precise and spatially consistent control, reducing the revisit drift by 66% over the strongest baseline. CNS further improves world consistency by reducing the revisit error. The same interface also works on backbones of different sizes (Team Wan, 2025; Chen et al., 2025; HaCohen et al., 2024) and transfers to out-of-distribution scenes. This suggests that the coordinate-based control interface is not tied to a particular architecture or training visual

domain. Together, our results show that representing action through world coordinates provides a simple yet effective alternative to conventional feature-based control.

To sum up, our contributions are as follows:

- We introduce **ActionRoPE**, a parameter-free action-control formulation for planar-camera world models that represents camera motion directly in world-coordinate rotary positions. It turns action control from learned feature conditioning into explicit coordinate alignment.
- Building on the coordinates exposed by ActionRoPE, we propose **Coordinate-Conditioned Noise Scheduling (CNS)**, which distinguishes geometrically known and newly revealed regions. It assigns token-wise denoising schedules according to this partition, with only a 0.015M-parameter input channel and no extra inference passes.
- We establish an evaluation protocol for action control covering motion fidelity, directional compliance, and closed-loop revisit. On the same backbone, our method outperforms four representative action-conditioning mechanisms and reduces the revisit drift by 66% over the strongest baseline. The same interface also transfers to backbones ranging from **1.3B** to **33B** parameters and to out-of-distribution scenes.

2 Related Work

2.1 Interactive Game World Models

Game world models have evolved from passive gameplay generation toward interactive visual simulation, where future observations are generated in response to user actions (Yu et al., 2025a; Liu et al., 2026). Genie (Bruce et al., 2024) learns latent actions from unlabeled gameplay, while subsequent diffusion-based models adopt explicit controls (Alonso et al., 2024), including keyboard inputs (Valevski et al., 2025; Decart & Etched, 2024), cross-scene control (Yu et al., 2025b), and real-time streaming generation (Zhang et al., 2025; He et al., 2025; Wang et al., 2026b). Recent work further incorporates richer states and interaction signals (Lin et al., 2026; Deng et al., 2026; Tong et al., 2026; Chen et al., 2026). Despite these differences, actions are typically injected as conditioning features, such as latent tokens, additive embeddings, language, adaptive normalization, or cross-attention (Bruce et al., 2024; Wang et al., 2026a; Zhu et al., 2026; AlayaWorld Team et al., 2026; Wang et al., 2026b). The resulting visual motion is therefore still learned from data.

Our work studies a setting where the action-to-motion relation is explicitly geometric. In planar-camera worlds, calibrated player actions determine planar camera displacement, allowing us to represent control directly through world coordinates rather than learned action features.

2.2 Geometry-Aware Positional Encoding

Rotary positional embeddings (RoPE) are widely used to encode spatio-temporal token coordinates in video DiTs (Su et al., 2024; Team Wan, 2025). Recent work has further incorporated camera geometry directly into positional representations rather than treating it only as an external condition (Miyato et al., 2024; Li et al., 2025). ReDirector (Park et al., 2025), ReRoPE (Li et al., 2026), SCoPE (Yin et al., 2026), and Matrix-Game 3.5 (Qian et al., 2026) encode camera transformations, viewing rays, or projection geometry into RoPE-like coordinates, improving geometric consistency under viewpoint changes. These methods, however, are designed for viewpoint control in general 3D scenes and take the camera pose or trajectory as input.

Our method adopts the same geometric view of positional encoding but applies it specifically to action control. Instead of encoding an external 3D camera pose, ActionRoPE directly converts the action-driven planar displacement into world coordinates, so that the same world location retains the same positional address throughout the camera motion.

2.3 Adaptive Diffusion Scheduling

Standard diffusion models apply a shared noise level across all tokens at each denoising step. Several works instead assign different noise levels to different tokens. Diffusion Forcing (Chen et al., 2024), Rolling Diffusion (Ruhe et al., 2024), FIFO-Diffusion (Kim et al., 2024), and streaming

generators (Chen et al., 2025; Sand.ai et al., 2025) vary noise levels across temporal positions, and recent image-generation methods further explore spatially adaptive schedules (Papa et al., 2026; Schusterbauer et al., 2026). Related inpainting approaches exploit known regions through explicit resampling or content injection (Lugmayr et al., 2022; Team Wan, 2025; HaCohen et al., 2024). These methods show that different regions can benefit from different denoising schedules, but the distinction is typically defined by time, an external mask, or explicit observed content.

In contrast, our method uses the world coordinates provided by ActionRoPE to define this distinction directly. Locations that are already covered by the initial observation are treated as geometrically known, while newly revealed locations still remain uncertain. CNS then uses this known–new structure to assign token-wise denoising schedules, allowing known regions to stabilize earlier without explicitly copying or reinserting any observed content.

3 METHOD

As shown in Figure 2, we propose ActionRoPE to turn a pretrained image-to-video diffusion transformer into an action-controlled world model without adding a control pathway. We first formalize planar-camera worlds, in which the effect of an action is a camera displacement (Section 3.1). We then write this displacement into the rotary positions, so that a token is addressed by where it is in the world rather than where it is on the screen (Section 3.2). The same coordinates also separate known regions from newly revealed ones. Building on this partition, we introduce Coordinate-Conditioned Noise Scheduling (CNS), which denoises known tokens earlier than newly revealed ones during both training and sampling (Section 3.3). Finally, Section 3.4 summarizes training and inference.

3.1 PLANAR-CAMERA WORLDS

We consider a class of interactive game worlds in which the dominant camera motion is approximately planar, including top-down and isometric games. Given an initial frame and an action sequence, each action moves the player. The camera then follows the player and the static world translates approximately rigidly in the image plane. Once the action-to-camera scale is calibrated, the visual effect of an action is therefore largely determined by geometry.

Existing action-conditioning mechanisms, however, generally treat actions as additional feature signals, whether as tokens, added features, modulated activations, or cross-attention inputs. Despite their different implementations, these approaches share the same formulation:

$$a_k \to \mathrm{Cond}(a_k) \to \text{Video Generator} \to \Delta_k, \tag{1}$$

where the model learns from data how an action should translate into the resulting visual motion. Consequently, both the direction and the magnitude of the motion are determined implicitly by the training distribution rather than explicitly by the control interface.

This formulation is unnecessarily indirect for planar-camera worlds. Here, $\Delta_k = (\Delta x_k, \Delta y_k)$ denotes the cumulative camera displacement at temporal cell k, i.e., the k-th latent frame of the video model. For a static world point, its screen-space location changes with Δ_k, but its location in the underlying world remains unchanged. Thus, rather than learning the effect of an action as a feature transformation, we seek to represent action-induced motion directly through the positional coordinates used by the video model. Our main observation is simple: *in planar-camera worlds, action control can be formulated as coordinate alignment*. This observation leads to ActionRoPE, which replaces screen-relative spatial positions with action-aligned world positions.

3.2 ACTION CONTROL AS WORLD COORDINATES

World-coordinate positional encoding. Consider a video DiT operating on temporal cell k and spatial token (i, j). Standard spatiotemporal RoPE assigns each token a position

$$\mathbf{p}_{kij} = (k, i, j), \tag{2}$$

where (i, j) denotes the spatial token location in the current image plane. Since these spatial coordinates are reset for every individual frame, the same world location in the scene receives different positional addresses over time whenever the camera moves.

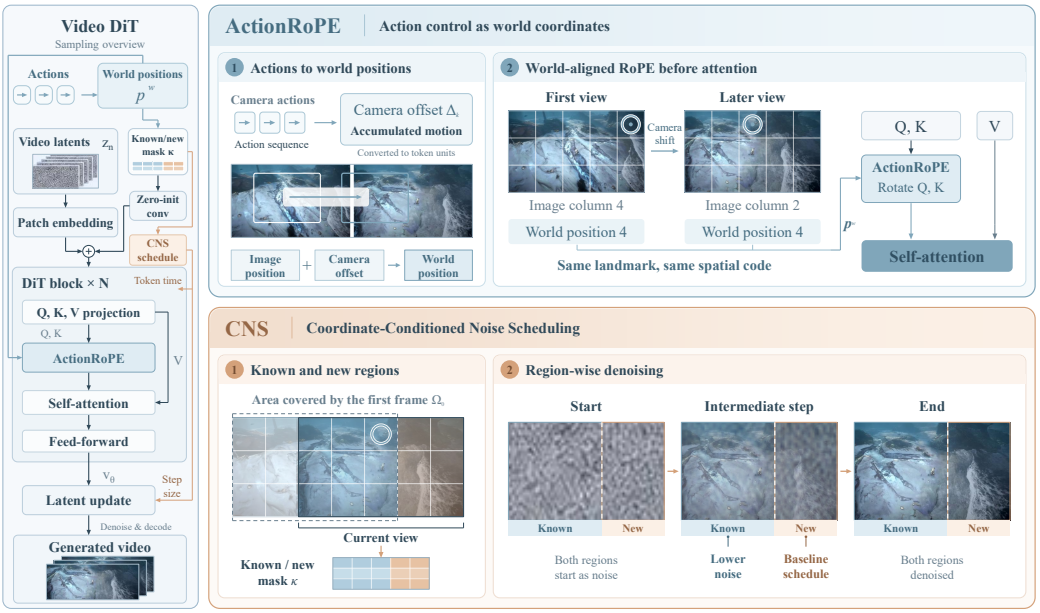

Figure 2: **Overview of ActionRoPE.** The left panel shows the video DiT, where ActionRoPE replaces the screen-relative RoPE before self-attention and the known/new mask κ is added to the patch embedding through a zero-initialized convolution. By binding camera motion to world coordinates, ActionRoPE keeps the same positional index for the same scene location across views, as illustrated in the upper-right panel. The same coordinates reveal which regions are known or newly revealed, and CNS assigns them different denoising schedules, as shown in the lower-right panel.

We instead define an action-aligned world coordinate for each token by explicitly compensating for the cumulative camera displacement Δ_k at temporal cell k:

$$\mathbf{p}^{w}_{kij} = \left(k,\ i + \frac{\Delta y_k}{s},\ j + \frac{\Delta x_k}{s}\right), \tag{3}$$

where s denotes the number of image pixels represented by one spatial token. In our Wan2.2 implementation, $s = 32$. The temporal coordinate remains unchanged.

We compute the rotary angles directly from the real-valued coordinates in Eq. 3. As RoPE naturally supports continuous positions, the camera displacement does not need to be quantized to the token grid. We refer to this parameter-free formulation as **ActionRoPE**.

The key property of ActionRoPE is that tokens corresponding to the same world location share the same spatial positional address across time. Suppose a static object appears at screen coordinate j in the first frame and shifts to $j - \delta$ after camera motion. Since the accumulated displacement $\Delta x_k / s$ increases by δ, both observations are assigned the same world coordinate, $(j - \delta) + \delta = j$. Cross-frame attention can therefore associate the two observations through positional alignment instead of inferring their relative displacement purely from visual appearance.

Known and newly revealed regions. World coordinates also allow us to determine whether a token corresponds to a region already observed in the first frame. Let Ω_0 denote the world-coordinate footprint covered by the first frame. We define a binary knownness field

$$\kappa_{kij} = \mathbb{I}\left[\mathbf{p}^{w}_{kij} \in \Omega_0\right]. \tag{4}$$

Tokens with $\kappa_{kij} = 1$ correspond to world locations already visible in the first frame, whereas tokens with $\kappa_{kij} = 0$ correspond to newly exposed regions that remain to be synthesized.

This partition is entirely geometric and is determined by the given action trajectory itself. It is therefore computed in the same way during training and inference and does not require any auxiliary

predictor. Besides determining the noise schedule in Section 3.3, κ is also given to the network as an input through a zero-initialized convolution added to the patch embedding. This adds only 0.015M parameters, leaves the pretrained model unchanged at initialization, and lets each token directly identify whether it belongs to a known or a newly revealed region.

3.3 Coordinate-Conditioned Noise Scheduling

ActionRoPE aligns previously observed world locations across time, and also exposes an asymmetry that standard diffusion and flow-matching schedules ignore. Tokens inside the first-frame footprint Ω_0 correspond to content that has already been observed, whereas tokens outside it correspond to genuinely unknown content. Standard schedules nevertheless apply the same noise level to every spatial location. We therefore propose **Coordinate-Conditioned Noise Scheduling (CNS)**, which adapts the denoising trajectory of each token according to its knownness κ, so that known regions settle first and newly revealed content is generated with reference to them.

Shift-based token-wise schedule. We build CNS on the shifted schedule of the underlying flow-matching model, $\sigma(u; S) = Su/\big(1 + (S-1)u\big)$, where $u \in [0, 1]$ is linear time and S the global shift. Given a global noise level σ, we recover its linear time $u = \sigma/\big(S - (S-1)\sigma\big)$ and assign every token its own shift and noise level,

$$S_{kij} = S\left(\frac{S_{\text{known}}}{S}\right)^{\kappa_{kij}}, \qquad \sigma_{kij} = \sigma(u; S_{kij}) = \frac{S_{kij}\, u}{1 + (S_{kij} - 1)\, u}, \tag{5}$$

with $S_{\text{known}} < S$. For newly revealed regions ($\kappa = 0$) we have $S_{kij} = S$ and CNS reduces exactly to the original schedule; for known regions ($\kappa = 1$) we have $S_{kij} = S_{\text{known}}$ and the token enters low-noise regimes earlier. All tokens share the endpoints $\sigma_{kij}(0) = 0$ and $\sigma_{kij}(1) = 1$, so they start from the same noisy initialization and converge to the same clean endpoint. Known regions simply stabilize earlier, and newly revealed regions are generated with reference to them.

Training with spatially varying noise. For a sampled global noise level σ, we noise each spatial location (k, i, j) with its own σ_{kij}, keeping the standard rectified-flow target and making the timestep embedding token-dependent:

$$\mathbf{x}_\sigma = (1 - \sigma_{kij})\, \mathbf{x}_0 + \sigma_{kij}\, \boldsymbol{\epsilon}, \qquad \mathbf{v}^* = \boldsymbol{\epsilon} - \mathbf{x}_0, \qquad t_{kij} = 1000\, \sigma_{kij}. \tag{6}$$

Following the standard image-to-video setup, the first temporal cell remains a clean anchor and is excluded from the loss. We additionally up-weight the loss on newly revealed regions, since their content is generated rather than retrieved through coordinate alignment.

Token-wise sampling. At inference, CNS requires no additional model evaluations. At step n the model is queried once with the token-wise timesteps, $\mathbf{v}_n = f_\theta\big(\mathbf{x}_n,\, t^{(n)}_{kij},\, \mathbf{p}^{w}_{kij},\, \kappa_{kij}\big)$, and each token advances along its own schedule,

$$\mathbf{x}_{n+1}(k, i, j) = \mathbf{x}_n(k, i, j) + \mathbf{v}_n(k, i, j)\, \big(\sigma^{(n+1)}_{kij} - \sigma^{(n)}_{kij}\big). \tag{7}$$

CNS thus changes only *when* each spatial token is denoised during sampling. It does not warp, copy, or paste any content from the first frame. The content of the known regions is still recovered through the world-coordinate correspondence that is established by ActionRoPE.

3.4 Training and Inference

During training, we fine-tune the backbone transformer with Eq. 6, taking Δ_k from the logged player positions and up-weighting the loss on newly revealed tokens by a factor of 2. Fine-tuning adapts the pretrained model to world-coordinate positions, while the action-to-motion mapping itself is given by geometry. At inference, the user's actions are integrated into Δ_k, which determines the world coordinates and κ, and the video is sampled with Eq. 7. For real-time use, we also distill it into a four-step causal student with a key–value cache. Since RoPE attention depends only on the relative positions of tokens, ActionRoPE carries over to this streaming setting.

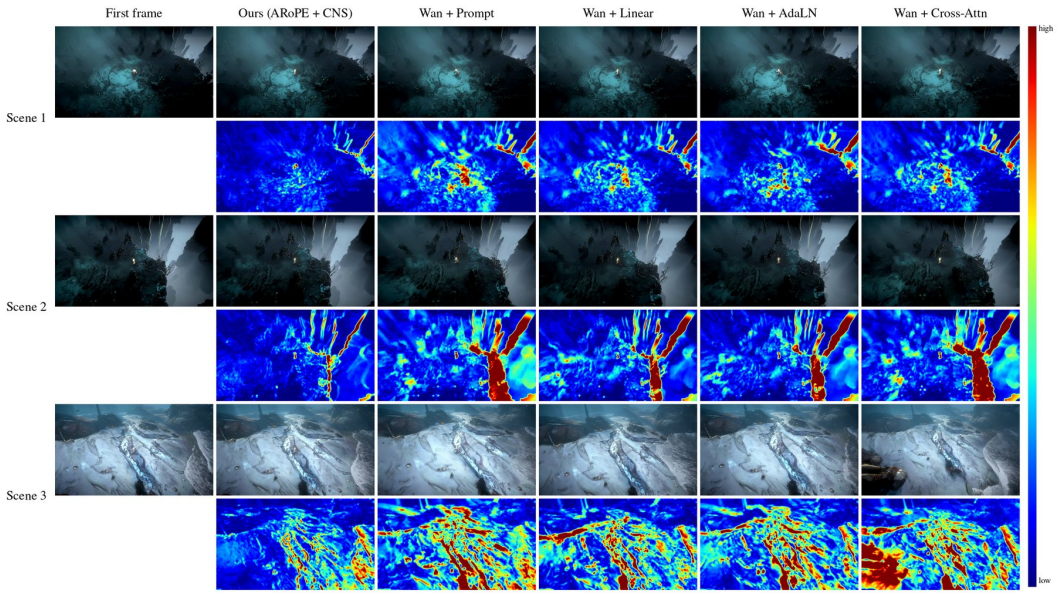

Figure 3: **Qualitative comparison of revisit consistency.** For each scene, the leftmost column shows the first frame. For each method, the top row shows the final frame after a full round trip, and the bottom row shows its absolute error against the first frame (blue indicates low and red indicates high error). Our method keeps the error low across the whole scene, while the four baselines show clearly larger errors that come from shifted or regenerated content.

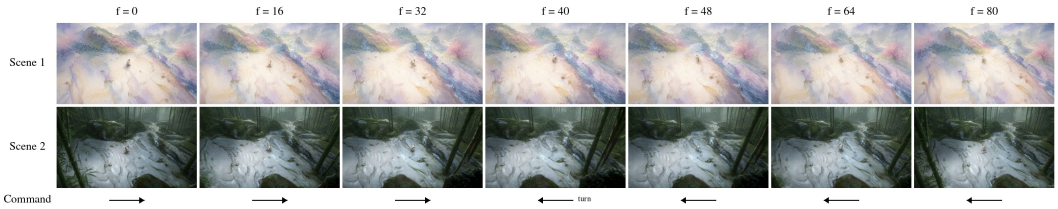

Figure 4: **Control on out-of-distribution first frames.** Each row starts from a first frame in a visual style that is absent from the training game (watercolor and bamboo forest, respectively), and the columns show the generated frames at increasing frame index f.

4 Experiments

4.1 Experimental Setup

Dataset. We build both training and evaluation on *No Rest for the Wicked*, a 2.5D action game rendered from a fixed top-down perspective. The camera follows the player, so the scene translates almost rigidly in the image plane. We record 50.9 hours of automated exploration and cut it into $22{,}782$ clips of 81 frames at 16 fps and 832×480 resolution. Of these, $18{,}528$ clips are used for training, and 982 held-out clips form the evaluation set. At test time, each model receives only the first frame of an evaluation clip and never its ground-truth continuation.

Backbones and training. We use Wan2.2-TI2V-5B (Team Wan, 2025) as our main backbone and implement ActionRoPE, CNS and all four baselines on it. All Wan models are fully fine-tuned from the same pretrained weights for $5{,}000$ steps with AdamW, a learning rate of 10^{-5} and a batch size of 16e. To examine whether our interface generalizes across architectures, we further change the backbone to SkyReels-V2-DF 1.3B (Chen et al., 2025), LTX-2.5 22B (HaCohen et al., 2024) and MiniMax-H3 33B (MiniMax, 2026) and evaluate under the same setting.

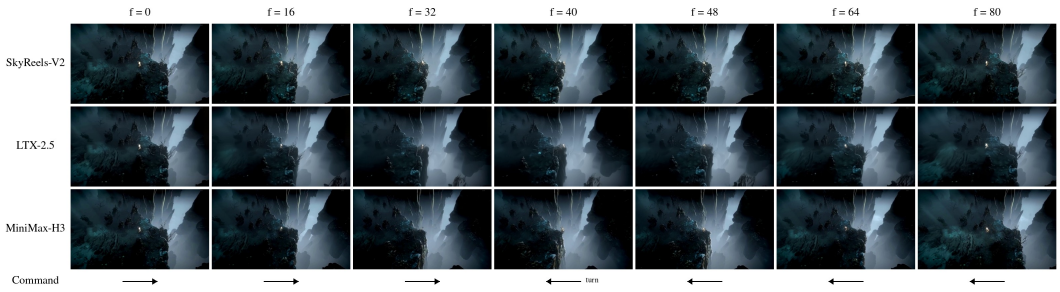

Figure 5: **Control on different backbones.** The same first frame and command are given to ActionRoPE on SkyReels-V2, LTX-2.5 and MiniMax-H3. The camera moves right and turns back at $f = 40$, and the columns show frames at increasing time.

Table 1: **Quantitative comparison results with baselines.** On the same Wan backbone, our method achieves the best motion fidelity, directional compliance and revisit consistency. It adds no parameters to the control path and only 0.015M parameters for the known/new mask channel.

		Motion fidelity		Direction	Revisit consistency				
Method	Control Params	slope	R^2	angle↓	PSNR↑	SSIM↑	NCC↑	DINO↑	drift↓
Wan + Prompt	0	0.911	−8.46	2.19	20.98	0.613	0.684	0.852	26.6
Wan + Linear	+0.18M	0.857	0.57	3.12	20.40	0.585	0.625	0.834	35.3
Wan + AdaLN	+66.5M	0.784	−0.52	1.92	21.11	0.604	0.685	0.862	22.6
Wan + Cross-Attn	+98.6M	0.798	−7.18	3.05	19.48	0.529	0.515	0.817	40.2
Ours	0	**0.981**	**0.952**	**0.68**	**23.94**	**0.691**	**0.831**	**0.892**	**7.8**

Baselines. We compare ActionRoPE with four representative action-injection mechanisms used by existing game world models, all reimplemented on the same Wan backbone for a fair comparison. **Prompt** describes the action in the text prompt of each frame (Zhu et al., 2026). **Linear** projects the action into a feature and adds it to the video tokens in every block (Wang et al., 2026a). **AdaLN** injects the action through adaptive layer normalization (AlayaWorld Team et al., 2026). **Cross-Attn** lets the video tokens attend to action tokens through additional cross-attention layers (Wang et al., 2026b). All four baselines receive the same per-frame camera displacement that ActionRoPE writes into the positional coordinates of the video tokens.

Metrics. For each evaluation clip, the model generates 81 frames from the first frame under a scripted action sequence. We measure the camera motion by matching SIFT features on the background, excluding a 160-pixel box around the player. *Motion fidelity* tests whether the camera moves by the commanded amount: on 100 first frames, we command straight walks at 0.5–$1.5\times$ the nominal speed, and we regress the measured displacement on the commanded one and report the slope and R^2, both ideally 1. *Directional compliance* tests whether it moves in the commanded direction: on all 982 first frames, we command a 5-second straight walk, and we report the median angular error. *Revisit consistency* tests whether the world stays fixed over time: we command a 2.5-second walk away followed by a return along the same path back to the start, and we compare the final frame with the first frame by PSNR, SSIM, normalized cross-correlation (NCC), DINOv2 feature similarity (Oquab et al., 2024) and residual geometric drift in pixels.

4.2 Qualitative Results

Figure 3 shows the revisit test on three evaluation scenes. For each method, we show the final frame of the round trip and its error map against the first frame. With ActionRoPE and CNS, the final frame closely matches the first frame, and the remaining error concentrates on thin structure edges. In contrast, the four baselines return with a noticeable offset: their error maps show the scene shifted from its original position, most clearly along the bright path in Scene 2, and Cross-Attn further generates a different layout in the lower-left of Scene 3.

Table 2: **Quantitative results across backbones.** With the same action interface, our method controls the camera on backbones ranging from 1.3B to 33B parameters. MiniMax-H3 generates 73 frames per clip, which are temporally resampled to 81 frames for evaluation.

Backbone	slope	R^2	angle↓	PSNR↑	SSIM↑	NCC↑	DINO↑	drift↓
SkyReels-V2-DF 1.3B	1.047	0.919	1.46	22.40	0.648	0.757	0.858	15.3
Wan2.2-TI2V 5B	0.981	0.952	0.68	23.94	0.691	0.831	0.892	7.8
LTX-2.5 22B	1.014	0.845	0.59	23.55	0.717	0.810	0.800	12.8
MiniMax-H3 33B	1.082	0.861	1.90	21.97	0.647	0.726	0.831	22.0

Table 3: **Ablation study.** We remove CNS from our full model on the Wan backbone, while keeping ActionRoPE in both compared models. Without CNS, all five revisit consistency metrics degrade, and the residual drift increases noticeably from 7.8 to 11.7 px.

Method	PSNR↑	SSIM↑	NCC↑	DINO↑	drift↓
Wan + ActionRoPE	22.81	0.652	0.788	0.885	11.7
Wan + ActionRoPE + CNS	**23.94**	**0.691**	**0.831**	**0.892**	**7.8**

Figure 4 shows control on first frames whose styles never appear in the training game. Given a command that moves right and then turns back at $f = 40$, the camera follows the path, and the newly revealed regions are filled in consistently with the style of the first frame.

4.3 Quantitative Results

Comparison with baselines. Table 1 compares our full model with the four baselines on the same backbone. Our model outperforms all baselines on every metric without adding any parameters to the control path. On revisit consistency, it improves over the strongest baseline on each metric by 2.8 dB in PSNR, 13% in SSIM and 21% in NCC, and reduces the residual drift by 66%.

Generalization across backbones. Table 2 reports the results of our full model on four backbones that differ by more than an order of magnitude in size, and Figure 5 shows the same first frame and command on three of these backbones. On all four backbones, the measured motion closely follows the commanded speed within 10% and the commanded direction within 2°, and the median revisit PSNR stays between 22.0 and 23.9 dB. This indicates that the control comes from the coordinate interface itself rather than from a particular choice of backbone.

4.4 Ablation Study

We further conduct ablation experiments on each component. Ablating ActionRoPE requires replacing it with another action-control mechanism, so the baselines in Table 1 can be viewed as ablations of ActionRoPE. Here, we therefore focus on the ablation of CNS. As shown in Table 3, adding CNS improves all revisit metrics, highlighting the effectiveness of CNS.

5 Conclusion

We presented ActionRoPE, a parameter-free action interface for planar-camera game world models. It writes action-driven camera motion directly into the rotary positions of video tokens, turning action control from a learned mapping into coordinate alignment. Building on the world coordinates, CNS lets known regions settle earlier, so that newly revealed content is generated with reference to them. Experiments show that our method follows the commanded motion more precisely and keeps the world more consistent on revisits than representative action-injection mechanisms. The same interface also transfers across backbones ranging from 1.3B to 33B parameters.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing and assist with parts of the code implementation. We also used an image generation model to create the 50 out-of-distribution first frames used for evaluation. We did not use generative AI tools to generate experimental results, analyze data, or formulate the main scientific claims. All AI-assisted text was reviewed and revised by the authors, and AI-assisted code was manually inspected, tested, and validated for correctness. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

AlayaWorld Team, Kaipeng Zhang, Chuanhao Li, Yifan Zhan, Yongtao Ge, Yuanyang Yin, Jiaming Tan, Kang He, Liaoyuan Fan, Mingliang Zhai, Ruicong Liu, Xiaojie Xu, Xuangeng Chu, Zhen Li, Zhengyuan Lin, Zhixiang Wang, Zian Meng, and Zihui Gao. AlayaWorld: interactive long-horizon world modeling – full technical report. *arXiv preprint arXiv:2607.18367*, 2026.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in Atari. *arXiv preprint arXiv:2405.12399*, 2024. NeurIPS 2024.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Danze Chen, Zeqing Wang, Ziyue Lin, Xingyi Yang, and Yeying Jin. H3-World: turning language understanding into world control. *arXiv preprint arXiv:2609.01560*, 2026.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, et al. SkyReels-V2: infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

Decart and Etched. Oasis: A universe in a transformer, 2024. `https://oasis-model.github.io/`.

Zhiyang Deng, Boran Zhang, Danze Chen, and Yeying Jin. WorldMind: decoupled game world model for state-aware NPC behavior. *arXiv preprint arXiv:2608.21439*, 2026.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. LTX-Video: realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Size Wu, Wei Li, Xuchen Song, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. FIFO-Diffusion: Generating infinite videos from text without training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Chunyang Li, Yuanbo Yang, Jiahao Shao, Hongyu Zhou, Katja Schwarz, and Yiyi Liao. ReRoPE: repurposing RoPE for relative camera control. *arXiv preprint arXiv:2602.08068*, 2026.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *arXiv preprint arXiv:2507.10496*, 2025. NeurIPS 2025.

Zijun Lin, Zeqing Wang, Cheston Tan, Bihan Wen, and Yeying Jin. StatePlay: state-aware game world models for mechanics-consistent generation. *arXiv preprint arXiv:2607.26754*, 2026.

Jiuming Liu, Chaojun Ni, Mengmeng Liu, Chensheng Peng, Fangjinhua Wang, Sitian Shen, Marc Pollefeys, Masayoshi Tomizuka, Ayush Tewari, and Per Ola Kristensson. Towards interactive video world modeling: Frontiers, challenges, benchmarks, and future trends. *arXiv preprint arXiv:2606.01164*, 2026.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

MiniMax. MiniMax-H3. `https://huggingface.co/MiniMaxAI/MiniMax-H3`, 2026.

Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. GTA: a geometry-aware attention mechanism for multi-view transformers. *arXiv preprint arXiv:2310.10375*, 2024. ICLR 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

Samuele Papa, Valentin De Bortoli, Guillaume Couairon, Daniel Sýkora, Romuald Elie, and Klaus Greff. AsyncPatch diffusion: spatially-flexible image generation. *arXiv preprint arXiv:2606.07079*, 2026.

Byeongjun Park, Byung-Hoon Kim, Hyungjin Chung, and Jong Chul Ye. ReDirector: creating any-length video retakes with rotary camera encoding. *arXiv preprint arXiv:2511.19827*, 2025.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models, 2025. `https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models/`.

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

Runjia Qian, Zile Wang, Jihai Zhang, Kai Zou, Wei Yu, Jiaxing Li, Zexiang Liu, Yaokun Li, Fei Kang, Kaichen Huang, Mengyin An, Haobo Zhang, Biao Jiang, Jiahua Wang, Haofeng Sun, Yang Liu, and Yangguang Li. Matrix-Game 3.5: Enhancing real-time streaming interactive world models with patch memory. *arXiv preprint arXiv:2608.29910*, 2026.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *International Conference on Machine Learning (ICML)*, 2024.

Sand.ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang, Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. MAGI-1: autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Johannes Schusterbauer, Ming Gui, Yusong Li, Pingchuan Ma, Felix Krause, and Björn Ommer. Denoising, fast and slow: Difficulty-aware adaptive sampling for image generation. *arXiv preprint arXiv:2604.19141*, 2026. CVPR 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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

Zizhao Tong, Hongfeng Lai, Zeqing Wang, Zhaohu Xing, Kexu Cheng, Haoran Xu, Zhao Pu, Shangwen Zhu, Ruili Feng, Jian Zhao, Yan Zhang, Hao Tang, Yeying Jin, and Ling Shao. SCOPE: simulating cross-game operations in playable environments for FPS world models. *arXiv preprint arXiv:2605.23345*, 2026.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations (ICLR)*, 2025.

Zeqing Wang, Danze Chen, Zhaohu Xing, Zizhao Tong, Yinhan Zhang, Xingyi Yang, and Yeying Jin. ReactiveGWM: steering NPC in reactive game world models. *arXiv preprint arXiv:2605.15256*, 2026a.

Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 3.0: Real-time and streaming interactive world model with long-horizon memory. *arXiv preprint arXiv:2604.08995*, 2026b.

Minghao Yin, Jiahao Lu, Wenbo Hu, Wang Zhao, Ying Shan, and Kai Han. SCoPE: sightline-coordinate positional encoding for video diffusion transformers. *arXiv preprint arXiv:2606.27345*, 2026.

Jiwen Yu, Yiran Qin, Haoxuan Che, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Position: Interactive generative video as next-generation game engine. *arXiv preprint arXiv:2503.17359*, 2025a.

Jiwen Yu, Yiran Qin, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. GameFactory: creating new games with generative interactive videos. *arXiv preprint arXiv:2501.08325*, 2025b.

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

Shangwen Zhu, Qianyu Peng, Zhao Pu, Zhilei Shu, Xiangrui Ke, Zhaohu Xing, Zizhao Tong, Zeqing Wang, Xinyu Cui, Huangji Wang, Jian Zhao, Yeying Jin, Fan Cheng, and Ruili Feng. Incantation: natural language as the action interface for multi-entity video world models. *arXiv preprint arXiv:2605.18601*, 2026.

Supplementary Material for ActionRoPE: Action as Coordinates for Spatially Consistent Control in Planar-Camera Game World Models

Contents

A Additional Qualitative Results **14**

A.1 Revisit Comparison . . . 14

A.2 Motion Scaling . . . 14

A.3 Out-of-Distribution Scenes . . . 14

A.4 Cross-Backbone Examples . . . 14

A.5 Effect of CNS . . . 14

B Additional Quantitative Results **19**

B.1 Out-of-Distribution Scenes . . . 19

B.2 Motion Scaling per Speed . . . 19

B.3 Noise Schedule Design . . . 19

B.4 Training versus Sampling . . . 19

C Implementation Details **19**

C.1 ActionRoPE and CNS . . . 19

C.2 Training and Sampling . . . 20

C.3 Baselines . . . 21

D Dataset **21**

E Evaluation Protocol **21**

F Limitations **22**

A Additional Qualitative Results

A.1 Revisit Comparison

Figures 6 and 7 show the revisit test of Figure 3 on six more evaluation scenes. All models receive the same command, moving right for 40 frames and then returning along the same path, and we show the final frame of each method with its error map against the first frame. With our method, the error stays low over most of the frame and concentrates on thin structure edges. The four baselines show larger high-error regions where the scene has not returned to its original position, for example on the bright ground in Scene 2 of Figure 6 and around the rock pillar in Scene 1 of Figure 7.

A.2 Motion Scaling

Figure 8 plots the measured displacement against the commanded displacement for all 500 motion-fidelity clips, i.e., 100 first frames at five speeds. Our method stays close to the ideal line at every speed. Linear and AdaLN receive the displacement as input and respond to the commanded speed, but they compress it, and their measured displacement levels off at higher speeds. Prompt and Cross-Attn receive only the direction of each action (Section C.3), so they move by nearly the same amount at every speed, which also explains their negative R^2 in Table 1.

A.3 Out-of-Distribution Scenes

Figure 9 repeats the revisit comparison on three first frames whose scenes do not appear in the training game. As on the evaluation set, the baselines return with larger errors than our method, most visibly on the pillars and in the lower right of Scene 2, while the lower part of Scene 3 changes for all methods, including ours. Figure 10 shows four more styles under the command of Figure 4. In each case, the camera follows the command and returns close to its starting view at $f{=}80$, and the regions revealed on the right are rendered in the style of the first frame.

A.4 Cross-Backbone Examples

Figures 11 and 12 apply the same first frame and command to all four backbones, including Wan2.2, which is not shown in Figure 5. All four backbones follow the rightward walk, turn back at $f{=}40$ and return close to the starting view at $f{=}80$. They differ mainly in how they render newly revealed regions, which appear darker for LTX-2.5 and MiniMax-H3 in Figure 11.

A.5 Effect of CNS

Figure 13 visualizes sampling with and without CNS for the same first frame, action and random seed. At several sampling steps, we decode the current clean-frame estimate of the last frame, where the region to the left of the orange line is covered by the first frame. In this example, CNS already recovers the floor tiles of the known region at step 6, and by step 14 the wall and door on the right have appeared, while the same region is still blurry without CNS. Figure 14 compares revisit error maps of our full model and the model trained without CNS on three evaluation scenes.

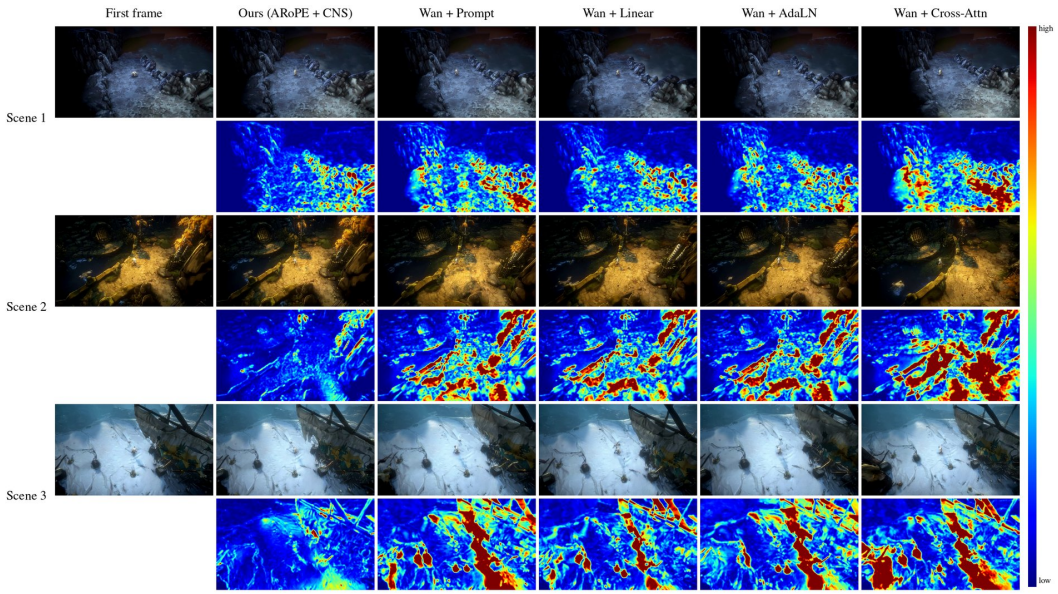

Figure 6: **More revisit comparisons (I).** For each scene, we show the final frame of each method after the round trip and its error map against the first frame.

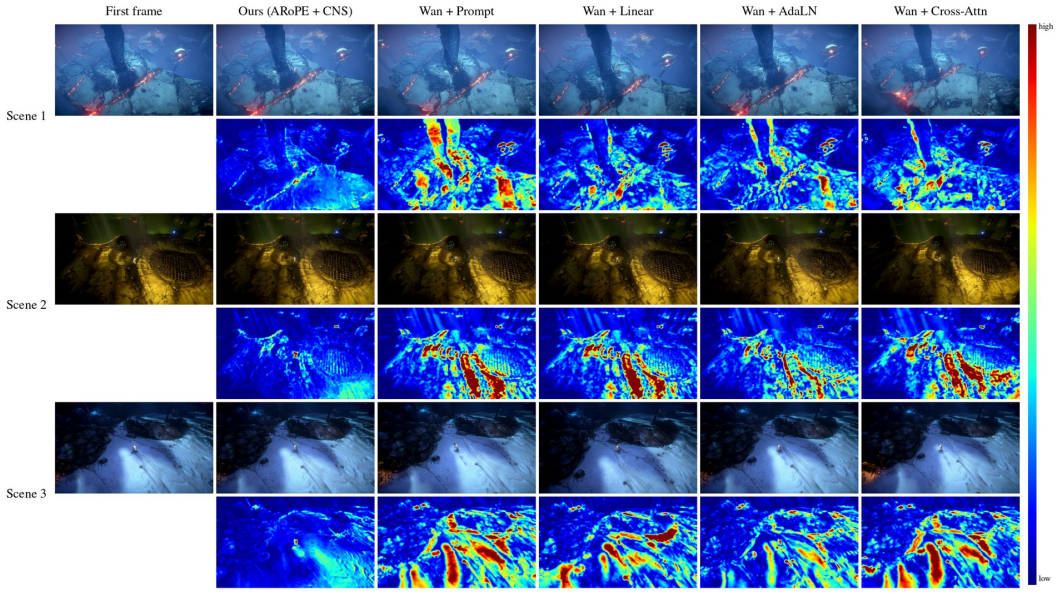

Figure 7: **More revisit comparisons (II).** Same layout as Figure 6, on three other evaluation scenes.

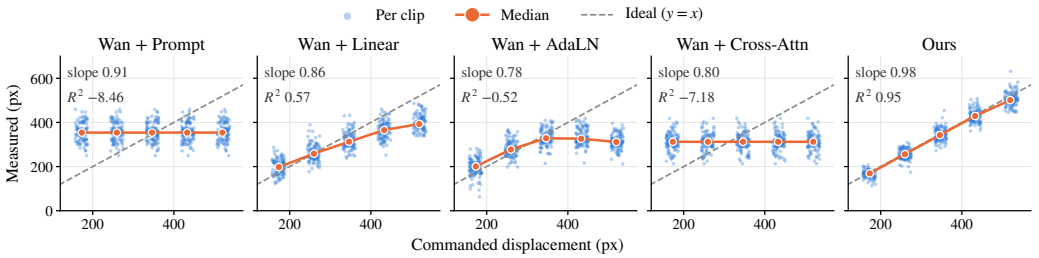

Figure 8: **Measured versus commanded displacement.** Each point is one clip, the orange line connects the medians at each speed, and the dashed line marks exact tracking. The slope and R^2 are those of Table 1.

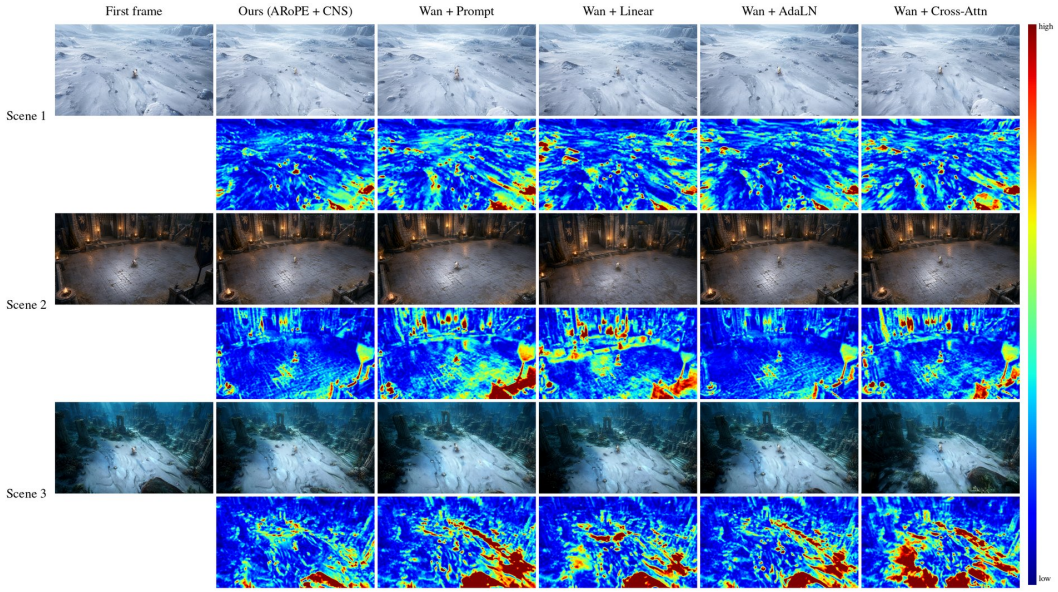

Figure 9: **Revisit comparison on out-of-distribution scenes.** The three first frames show an arctic tundra, a medieval castle and an underwater ruin. Same layout as Figure 6.

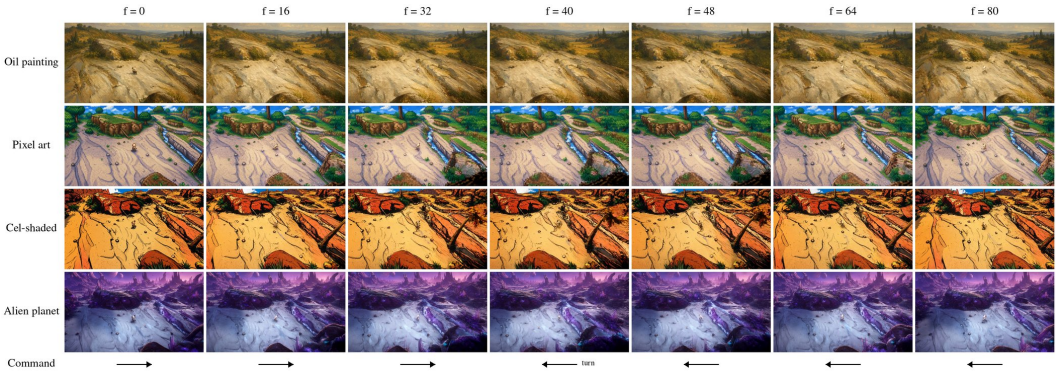

Figure 10: **More out-of-distribution styles.** Our method on four first frames with styles that never appear in the training game, under a command that moves right and turns back at $f{=}40$.

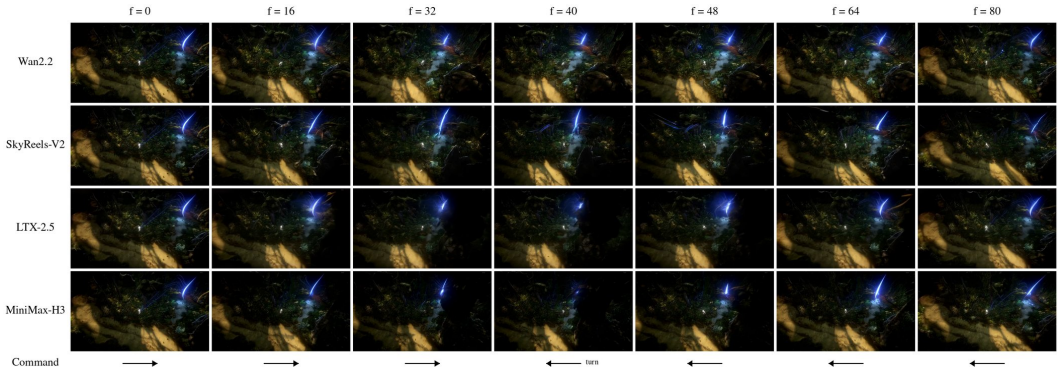

Figure 11: **Cross-backbone example (I).** The same first frame and command on the four backbones.

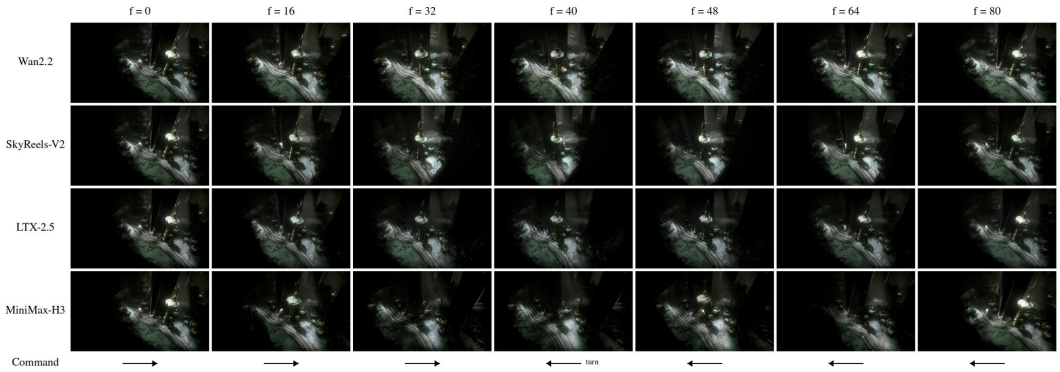

Figure 12: **Cross-backbone example (II).** Same setting as Figure 11, on another evaluation scene.



Figure 13: **Sampling with and without CNS.** Clean-frame estimates of the last frame during sampling, under a straight walk to the right. The region to the left of the orange line is covered by the first frame.

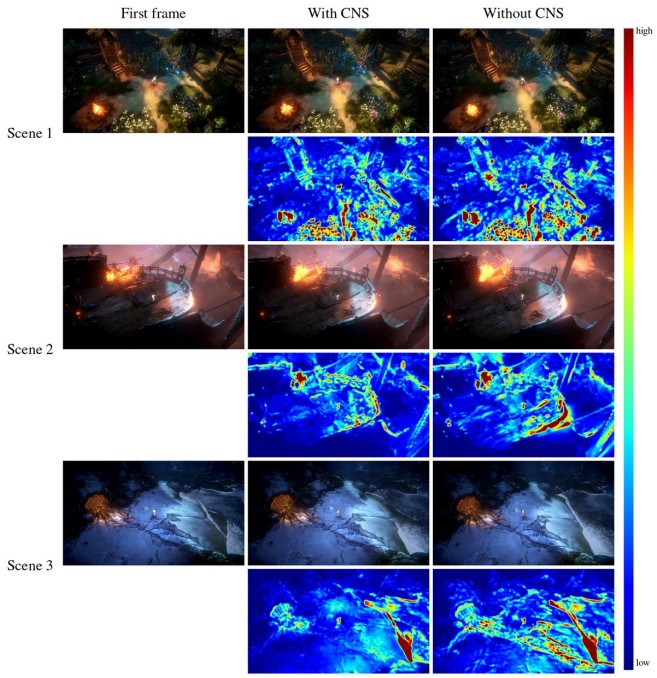

Figure 14: **Revisit with and without CNS.** Final frames after the round trip and their error maps, for our full model and the model trained without CNS.

B Additional Quantitative Results

B.1 Out-of-Distribution Scenes

Table 4 evaluates all methods on the 50 out-of-distribution first frames with the protocol of Section E, where motion fidelity uses 10 of these first frames at five speeds. Revisit PSNR and SSIM are lower than on the evaluation set for all methods, but our method remains the best on every metric, and its residual drift (7.6 px) is less than half that of the strongest baseline (16.5 px).

B.2 Motion Scaling per Speed

Table 5 lists the median ratio between the measured and commanded displacement at each speed, i.e., the medians in Figure 8. Our ratios stay between 0.96 and 0.99. The baselines overshoot slow commands and undershoot fast ones. For Prompt and Cross-Attn, the ratio falls in inverse proportion to the commanded speed, since their measured displacement does not change with it.

B.3 Noise Schedule Design

Besides the shift-based schedule of Eq. 5, we test a power-law schedule $\sigma_{kij} = \sigma^{1+\alpha\kappa_{kij}}$, which also keeps newly revealed tokens on the original schedule. Figure 15 shows the resulting noise level of a known token for each variant, and Table 6 reports the results on Wan2.2. Every coordinate-conditioned schedule reaches a higher revisit PSNR than the uniform schedule in this comparison. We use the shift-based form because it reuses the shifted schedule of each backbone and needs a single hyperparameter, S_{known}.

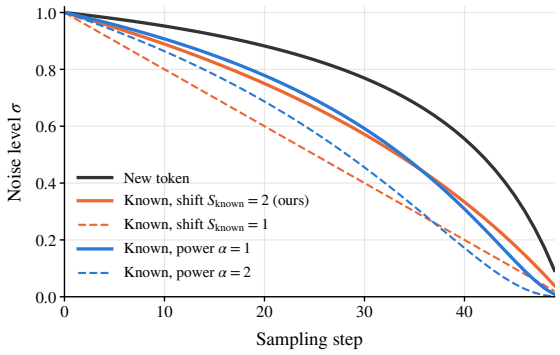

Figure 15: **Noise level during sampling.** A newly revealed token follows the original schedule of Wan2.2 (S=5), while a known token (κ=1) follows a lower curve under each CNS variant.

B.4 Training versus Sampling

CNS changes both the noise applied during training and the schedule used during sampling. To separate the two, we cross the training schedule with the sampling schedule. Sampling a model trained without CNS with the CNS schedule leaves revisit PSNR unchanged (22.82 vs. 22.81 dB), and sampling the CNS-trained model with the uniform schedule keeps its revisit PSNR (24.06 vs. 23.94 dB). The effect of CNS therefore comes from training rather than from the sampler alone.

C Implementation Details

C.1 ActionRoPE and CNS

Table 7 summarizes the settings of each backbone. The displacement Δ_k is averaged over the video frames that form latent frame k, and s is the number of pixels covered by one spatial token after the VAE and the patch embedding. The world coordinates of Eq. 3 are added to the spatial

Table 4: **Comparison on out-of-distribution scenes.** Same metrics as Table 1, on 50 first frames whose scenes and styles do not appear in the training game.

	Motion fidelity		Direction	Revisit consistency				
Method	slope	R^2	angle↓	PSNR↑	SSIM↑	NCC↑	DINO↑	drift↓
Wan + Prompt	0.831	−2.39	4.66	17.64	0.360	0.664	0.912	30.3
Wan + Linear	0.814	0.470	2.60	16.64	0.359	0.605	0.883	47.7
Wan + AdaLN	0.762	−0.49	1.71	18.44	0.376	0.721	0.921	16.5
Wan + Cross-Attn	0.762	−2.90	5.41	17.66	0.363	0.635	0.902	36.0
Ours	**0.922**	**0.926**	**0.94**	**19.67**	**0.417**	**0.809**	**0.928**	**7.6**

Table 5: **Measured-to-commanded displacement ratio per speed.** Median over 100 first frames of the evaluation set. The ideal value is 1, and the closest value in each column is in bold.

Method	0.5×	0.75×	1.0×	1.25×	1.5×
Wan + Prompt	2.05	1.36	1.02	0.82	0.68
Wan + Linear	1.14	**1.00**	0.90	0.84	0.76
Wan + AdaLN	1.16	1.07	0.95	0.75	0.60
Wan + Cross-Attn	1.80	1.20	0.90	0.72	0.60
Ours	**0.98**	0.99	**0.99**	**0.99**	**0.96**

Table 6: **Noise schedule variants on Wan2.2.** Revisit consistency and direction error on the evaluation set.

Schedule	PSNR↑	drift↓	angle↓
Uniform (without CNS)	22.81	11.7	**0.67**
Power, α=1	23.65	9.1	0.71
Power, α=2	23.57	9.5	0.77
Shift, S_{known}=1	23.20	10.7	0.87
Shift, S_{known}=2 (ours)	**23.94**	**7.8**	0.68

RoPE positions of each backbone, and for MiniMax-H3, whose spatial positions are not spaced by one, Δ_k/s is scaled by the same spacing. The knownness κ is computed on the latent grid; for the token-wise timesteps, it is averaged over the latent positions of each token, so tokens on the footprint boundary receive intermediate values. The mask channel is a zero-initialized layer that maps κ to the hidden width and adds it to the patch embedding, i.e., a single-channel convolution with the patch-embedding kernel on Wan2.2 and SkyReels-V2, and a linear layer on LTX-2.5 and MiniMax-H3. For CNS, S is the default sampling shift of each backbone, and we keep the ratio $S_{\text{known}}/S = 0.4$ found on Wan2.2 for the other backbones.

Table 7: **Per-backbone settings.** s: pixels per spatial token. Mask: parameters of the known/new mask channel. S, S_{known}: CNS shifts. CFG: classifier-free guidance scale at sampling.

Backbone	s	Mask	S	S_{known}	Fine-tuning	LR	Batch	Steps / CFG
SkyReels-V2-DF 1.3B	16	7,680	5	2	full	10^{-5}	16	50 / 1
Wan2.2-TI2V 5B	32	15,360	5	2	full	10^{-5}	16	50 / 1
LTX-2.5 22B	32	8,192	7.77	3.1	LoRA, rank 128	10^{-4}	16	30 / 3
MiniMax-H3 33B	32	10,752	12	4.8	LoRA, rank 128	10^{-4}	8	50 / 1

C.2 Training and Sampling

All models start from the public pretrained weights and are trained for 5,000 steps with AdamW, a weight decay of 0.01, a linear warm-up followed by a constant learning rate, bf16 precision and gradient clipping at 1.0. The flow-matching loss excludes the first latent frame, and for our models, newly revealed tokens receive a weight of 2. Text prompts describe only the scene, using one of

18 scene descriptions of the game, and the scene description is dropped with probability 0.1. At evaluation, all models use this scene-free prompt, the same first frames and commands, and a fixed random seed. Sampling uses the number of steps and guidance scale in Table 7, with one model evaluation per step except for the guided LTX-2.5.

C.3 BASELINES

All baselines are built on the same Wan2.2 backbone with standard RoPE and no mask channel, and are trained with the same data, learning rate, batch size, number of steps and seed as our model. All of them are driven by the same per-frame camera displacement as our model (Section 4.1), which each baseline converts into its own action format; Prompt and Cross-Attn quantize it into discrete direction labels and therefore keep the direction but not the speed. Since they have no known/new partition, they use a uniform loss weight. Newly added layers are zero-initialized, so each model starts from the pretrained backbone.

- **Prompt** (Zhu et al., 2026) appends the action of each latent frame to its text prompt, as one of nine labels (idle and eight directions), and the tokens of each latent frame attend only to their own prompt. It adds no parameters and receives the direction but not the speed.
- **Linear** (Wang et al., 2026a) maps the cumulative displacement Δ_k/s through one bias-free linear layer per block and adds the result to the tokens of latent frame k at the input of each of the 30 blocks (0.18M parameters).
- **AdaLN** (AlayaWorld Team et al., 2026) encodes the cumulative displacement and its per-frame increment with sinusoidal features, maps them with an MLP, and adds the result to the timestep modulation of every block, per latent frame (66.5M parameters).
- **Cross-Attn** (Wang et al., 2026b) follows the keyboard branch of Matrix-Game 3, where each latent frame is represented by a four-key vector (up, down, left, right) and the video tokens attend to these action tokens along time in the first 15 blocks, between the text cross-attention and the feed-forward layer (98.6M parameters). Like Prompt, it receives the direction but not the speed.

D DATASET

The recordings consist of 50.9 hours of automated exploration in two sessions, one with the in-game day–night cycle and weather (34.55 h) and one with fixed daytime and clear weather (16.39 h). The engine logs the player position at 20 Hz, and we map it to screen displacement with a calibrated linear map that also accounts for terrain height. We fit the scale of this map against the background motion measured in the videos (R^2 of 0.99 horizontally and 0.89 vertically). As a check on the action labels, the eight-way movement direction predicted from the logs agrees with that measured by optical flow on 87.7% of 42,456 samples, and on 98.0% within one neighboring direction.

We remove corrupted segments, teleport windows, periods in which the player is stuck for at least 2 s, and low-light frames, leaving 33.35 usable hours. These are cut into 20,882 clips of 81 frames with a stride of 84 frames and resized to 832×480. The dataset further contains 1,900 transition clips that splice two scenes. Since adjacent clips overlap heavily, we split training and evaluation by entire exploration segments, and exclude 16 buffer segments from both. From the 20,983 clips of the training split, we remove one clip with invalid labels, 257 transition clips whose source clips lie in the evaluation split, and 2,197 clips in which the game world runs slower than 0.8× real time, which leaves 18,528 training clips. The evaluation set contains the 982 held-out clips with valid labels that run at least at 0.9× real time.

E EVALUATION PROTOCOL

Commands. The nominal speed (1.0×) is the median walking speed in the training data, 4.33 px per frame horizontally. Motion fidelity uses the first 100 evaluation first frames and commands a straight walk to the right for 80 frames at 0.5, 0.75, 1.0, 1.25 and 1.5× this speed. Directional compliance commands the same walk at 1.0× on all 982 first frames. Revisit consistency commands

a walk to the right for 40 frames followed by a walk to the left for 40 frames, so that the commanded camera position at the final frame equals the initial one.

Measuring motion. We detect SIFT features on contrast-enhanced grayscale frames, excluding a 160×160-pixel box at the screen center where the player stays, and fit an affine motion with RANSAC. The displacement between the first and the last frame is measured directly, and through intermediate frames when direct matching fails. Clips whose motion cannot be measured, mostly very dark first frames, are excluded from the medians.

Metrics. For motion fidelity, we regress the measured displacement on the commanded one through the origin and report the slope. R^2 is computed around the mean of the measured displacement, so it becomes negative when the measured displacement does not vary with the command. For directional compliance, we report the median absolute angle between the measured and commanded displacement. For revisit consistency, we compare the final frame with the first frame outside the central box. Before computing PSNR, we match the mean and standard deviation of each color channel of the final frame to the first frame, so that global brightness changes are not counted. SSIM uses an 11×11 Gaussian window with $\sigma=1.5$ on grayscale frames, and NCC is computed on grayscale frames. DINO is the cosine similarity between the DINOv2-base class tokens of the two frames, with the central box filled with gray. The residual drift is the magnitude of the measured displacement between the first and the final frame.

Out-of-distribution first frames. The 50 out-of-distribution first frames depict scenes and art styles that do not appear in the game, such as cyberpunk cities, deserts, oil paintings and pixel art, each with the player at the screen center. They are evaluated with the same commands and metrics, with motion fidelity on the first 10 of them.

F LIMITATIONS

ActionRoPE assumes that the static world translates approximately rigidly in the image plane, which holds for top-down and isometric cameras that follow the player. It does not model zoom or strong parallax, and moving objects do not follow the world coordinates. The knownness κ covers only the first-frame footprint, so regions that become visible later are not treated as known when the camera returns to them.