ACTIONSPLICE: IN-FLIGHT ACTION EDITING FOR INTERACTIVE WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Chunk-autoregressive video world models typically generate each chunk under one action. When an action changes during sampling, waiting until the next chunk delays the response, while directly switching the conditioning leaves the intermediate solver state shaped by the previous action. Restarting sampling under the revised action avoids this mismatch but repeats completed computation. We introduce *ActionSplice*, which edits the interrupted state through Counterfactual State Transport (CST). A lightweight corrector moves the interrupted solver state toward the matched counterfactual solver state induced by the revised action at the same solver step. The world model and sampler remain frozen, and sampling resumes without replaying completed evaluations. CST_R retargets the entire active chunk, while CST_T preserves the temporal prefix at the intervention step and corrects only the suffix. On minWM–Wan Action2V and HY-WM1.5, we measure fidelity against matched rollback references. Relative to condition swapping, CST_R reduces LPIPS by 61.5% and 75.9%, and CST_T reduces suffix LPIPS by 56.1% and 77.5%, respectively.

1 INTRODUCTION

Interactive video world models must generate frames quickly and respond promptly to changes in control. Recent diffusion world models support action-conditioned visual simulation and real-time streaming generation (Valevski et al., 2025; Wang et al., 2026; Zhao et al., 2026a). However, high frame throughput does not guarantee a short delay between an action request and its visible effect. We consider chunk-autoregressive generation in which each chunk is initially sampled under one action through K solver evaluations. If a revised action arrives after evaluation $r < K$, the active chunk is only partially generated, but its intermediate state already reflects the previous action. Waiting until the next chunk postpones the response. We study *in-flight action editing*: incorporating the revised action into the active chunk without replaying completed solver evaluations or modifying the committed history of completed chunks.

Direct condition swapping changes conditioning for subsequent solver evaluations but does not modify the representation at the moment of the switch. We refer to this mismatch as *state misalignment*. Full rollback resolves it by restarting the active chunk under the revised action, but repeats the first r solver evaluations. The representation reached at step r provides a counterfactual target under matched initial conditions and committed history. We seek to approximate this target from the interrupted representation, allowing sampling to continue without repeating completed evaluations.

We introduce *ActionSplice*, an inference framework built around Counterfactual State Transport (CST). A lightweight corrector trained on matched rollback pairs predicts a masked residual at the interruption step, producing a resumable solver state. The frozen world model then performs the remaining $K - r$ evaluations. The retargeting variant CST_R applies the revised action to the entire active chunk. The temporal splicing variant CST_T supports a scheduled action transition at a temporal boundary m within the chunk. It preserves prefix coordinates of the interrupted solver state at the intervention step and restricts the correction to the suffix. The solver step r specifies when sampling is interrupted. The temporal boundary m specifies where the action changes in the generated sequence. Full rollback constructs training targets and evaluation references. It is not executed during ActionSplice inference.

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.

We evaluate CST on minWM–Wan Action2V (Zhao et al., 2026a) and HY-WM1.5 (HunyuanWorld, 2025), with a separate corrector for each backbone and variant. We examine receipt steps, directed action transitions, temporal boundaries, and repeated interruptions. We also evaluate camera trajectories and human judgments of action following and response delay. Fidelity is measured against matched full rollback for CST_R and matched rollback with prefix clamping for CST_T. Relative to condition swapping, CST_R reduces LPIPS by 61.5% on minWM and 75.9% on HY-WM1.5, while CST_T reduces suffix LPIPS by 56.1% and 77.5%, respectively. Compared with waiting, CST_R achieves pixel-ready speedups of $2.69\times$ and $1.64\times$ on minWM and HY-WM1.5, respectively. On the HY-WorldPlay benchmark, CST_R obtains a PSNR of 25.66 against the original rollout.

2 Related Work

Interactive video world models. Diffusion-based world models support action-conditioned visual simulation in interactive game environments (Alonso et al., 2024; Valevski et al., 2025). Recent systems extend these capabilities to streaming generation with camera control. WorldPlay (Sun et al., 2025) uses dual action representations and reconstituted context memory to maintain geometric consistency over long rollouts. Matrix-Game 3.0 (Wang et al., 2026) combines long-horizon memory with multi-segment autoregressive distillation. HY-World 1.5 (HunyuanWorld, 2025) targets real-time interactive generation with geometric consistency. minWM (Zhao et al., 2026a) and BiWM (Rui et al., 2026) provide frameworks for adapting bidirectional video diffusion backbones into autoregressive world models with camera control. Other work addresses the training and efficiency of autoregressive generators. Causal Forcing (Zhu et al., 2026), Causal Forcing++ (Zhao et al., 2026b), and Causal-rCM (Zheng et al., 2026) develop training and distillation methods for efficient causal generation. Self Forcing (Huang et al., 2026) addresses exposure bias by training on histories generated by the model itself. ActionSplice corrects the active solver state when an action update arrives during sampling.

Interactive condition changes and memory. Prior work addresses responsiveness to changing conditions through training objectives and memory management during inference. Delta Forcing (Wu et al., 2026b) constrains teacher supervision within an adaptive trust region during training to balance event responsiveness and temporal consistency. Echo-Forcing (Wu et al., 2026a) separates stable, recent, and recalled memory to support prompt switching and scene recall. LongLive (Yang et al., 2025) refreshes cached states after prompt switches, while Anchor Forcing (Yang et al., 2026) reconstructs the cache from anchor memories. Visko Orbis (Gao et al., 2026) supports live prompt updates during streaming video generation. ActionSplice focuses on camera or action updates received while the active chunk is being sampled. CST estimates the counterfactual solver state that matched rollback would reach at the same solver step.

Inference-time reuse and acceleration. Video diffusion methods reduce inference cost through caching and sparse computation. TeaCache (Liu et al., 2025) and FasterCache (Lyu et al., 2025) reuse features across denoising steps, while Pyramid Attention Broadcast (Zhao et al., 2025) reuses attention outputs. Sparse VideoGen (Xi et al., 2025) and SparsePR (Taghavi et al., 2026) reduce attention cost through spatiotemporal sparsity. Light Interaction (Lu et al., 2026) combines context selection, denoising reuse, and sparse attention for interactive generation. X-Cache (Zeng et al., 2026) and C^3ache (Zhao et al., 2026c) reuse computation across chunks. Chorus (Liu et al., 2026) reuses intermediate information across similar requests while maintaining alignment with their prompts. ActionSplice corrects the active solver state toward its matched counterfactual at the interruption step, then resumes sampling without replaying completed evaluations.

Revisable denoising and editing of intermediate states. Diffusion Forcing (Chen et al., 2024) assigns independent noise levels to tokens, while Diffusion ReRoll (Kim et al., 2026) selectively renoises stable regions to enable revision across a temporal horizon. For source editing, SDEdit (Meng et al., 2021) uses re-noising, EDICT (Wallace et al., 2023) uses inversion, and Layered Diffusion Brushes (Gholami & Xiao, 2025) caches latents for localized edits. FateZero (Qi et al., 2023) reuses attention maps from video inversion to preserve structure and temporal consistency. ActionSplice operates on an unfinished action-conditioned rollout. CST learns from matched rollback pairs to approximate the solver state induced by the requested intervention at the current step. Sampling resumes from the corrected state without restarting or replaying completed evaluations.

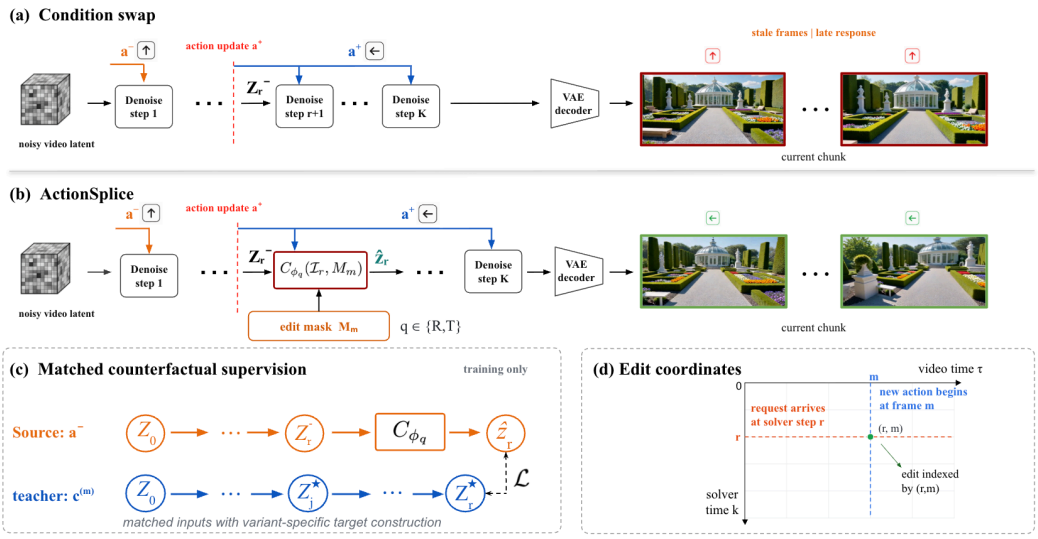

Figure 1: ActionSplice overview. (a) Condition swapping retains the interrupted representation. (b) CST predicts a masked correction at receipt step r and resumes sampling. (c) Matched rollback supplies training targets, with prefix clamping for CST_T. (d) The receipt step r indexes solver time and the boundary m indexes latent video time.

3 Method

ActionSplice corrects the interrupted solver state toward its matched counterfactual. The frozen world model then resumes sampling without replaying completed evaluations (Figure 1).

3.1 Problem Formulation

A frozen world model generates a chunk with T temporal latent positions through K solver evaluations, initially conditioned on action a^-. An update arrives after $r \in \{1, \dots, K-1\}$ evaluations, requesting a revised action a^+. Previously completed chunks form the committed history and remain unchanged. Let z_r^- denote a resumable representation of the interrupted solver state at step r.

The request specifies a boundary $m \in \{0, \dots, T-1\}$, the first latent position assigned the revised action. For each position i, the requested action assignment $c^{(m)}$ and correction mask M_m are

$$c_i^{(m)} = \begin{cases} a^-, & i < m, \\ a^+, & i \geq m, \end{cases} \qquad M_m[i] = \mathbf{1}[i \geq m]. \tag{1}$$

The receipt step r specifies when sampling is interrupted, while m specifies where the requested action changes within the chunk. CST_R retargets the entire chunk with $m = 0$. CST_T requests the revised action in the suffix with $0 < m < T$, retaining the previous action assignment in the prefix. We incorporate the request into the active chunk without replaying completed evaluations.

3.2 Matched Counterfactual Targets

Let $z_r^\star$ denote the matched counterfactual state representation after evaluation r. To construct it, the source and target branches share the prompt, initial state, committed history, solver schedule, and stochastic inputs when applicable. The source branch runs the first r evaluations under a^-. For CST_R, the target branch runs the same evaluations under a^+ to obtain $z_r^\star$.

For CST_T, the target branch runs under the mixed action assignment $c^{(m)}$ while enforcing a prefix constraint after each evaluation. Target prefix quantities are replaced by the corresponding source quantities from the same solver step, while the suffix retains the target branch values. Consequently,

$$z_r^\star[i] = z_r^-[i], \qquad i < m.$$

Appendix A.1 specifies the clamped quantities, conditioning, and operation order for each implementation. Rollback is used only to construct training targets and evaluation references, not during ActionSplice inference.

3.3 Counterfactual State Transport

The corrector input $\mathcal{I}_r$ contains z_r^-, the initial state, preceding chunk, old and revised action conditioning, receipt step, and retained sampling information. Appendix A.1 specifies the input encoding. For edit type $q \in \{R, T\}$, the corrector C_{ϕ_q} with learned parameters ϕ_q predicts a residual that is restricted by the temporal mask:

$$\widehat{z}_r = z_r^- + M_m \odot C_{\phi_q}(\mathcal{I}_r, M_m), \tag{2}$$

where $\odot$ denotes elementwise multiplication. The mask broadcasts over channel and spatial dimensions, ensuring $(\mathbf{1} - M_m) \odot (\widehat{z}_r - z_r^-) = 0$. For CST_T, this preserves prefix coordinates at the intervention step but does not guarantee an identical decoded prefix after further sampling.

Sampling resumes from $\widehat{z}_r$ under conditioning constructed from $c^{(m)}$. Appendix A.1 specifies scheduler conversion, prefix handling, and cache operations. Each intervention uses one corrector call and the remaining $K-r$ solver evaluations, whereas full rollback performs K solver evaluations after receipt. The corrector, state conversion, and cache updates add overhead beyond this count.

3.4 Corrector Training

We train a separate corrector for each backbone and intervention type. Each corrector is a six-block residual 3D encoder-decoder conditioned on action inputs and receipt step through FiLM (Perez et al., 2018). We optimize only the corrector parameters ϕ_q. The world model and sampler remain frozen, and the mask M_m is fixed by the supplied boundary m. We minimize masked normalized mean squared error:

$$\mathcal{L} = \frac{\|M_m \odot (\widehat{z}_r - z_r^\star)\|_2^2 / N_m}{\max(\|M_m \odot z_r^\star\|_2^2 / N_m, \epsilon)}, \tag{3}$$

where N_m counts editable scalar elements after broadcasting the mask and $\epsilon = 10^{-8}$. Appendix A.1 specifies the input encoding and corrector architecture.

Training pairs are collected from recurrent rollouts whose committed histories include outputs from earlier corrections. At each interruption, the source and target branches use the same committed history. Training uses stored capture pairs, so gradients do not propagate through the rollout that produced this history. Appendix A.3 details initialization and checkpoint selection.

4 Experiments

4.1 Experimental Setup and Evaluation Protocol

We evaluate ActionSplice on minWM–Wan Action2V (Zhao et al., 2026a) and HY-WM1.5 (HunyuanWorld, 2025), using four solver evaluations per chunk. We train a separate corrector for each backbone and CST variant. Both backbones share a prompt-disjoint split of 120 training, 30 validation, and 30 test scenes. Training captures come from recurrent rollouts whose committed histories include earlier corrections. Checkpoint selection uses the validation set. The main evaluation uses receipt step $r = 2$, with 30 matched groups of test prompts for CST_R and 90 matched combinations of test prompts and temporal boundaries for CST_T, covering $m \in \{1, 2, 3\}$ for each prompt. Transitions cover all six directed pairs among forward, backward, and yaw left motion. Additional experiments vary the receipt step, action transition, temporal boundary, and number of interruptions.

We compare CST with waiting, condition swapping, partial rollback, re-noising, and matched rollback. Waiting applies the revised action in the next chunk. Condition swapping changes the conditioning for the remaining evaluations without correcting the interrupted state representation. Partial rollback restores the preceding solver checkpoint and repeats one completed evaluation. Re-noising perturbs a clean estimate to the noise level preceding the final two evaluations and runs them under the revised action. The reference is full rollback for CST_R and rollback with prefix clamping for

Table 1: Whole-chunk retargeting at $r = 2$ of $K = 4$ solver evaluations over 30 test prompts. Fidelity is measured over the active chunk against matched full rollback. Bold and underlined values mark the best and second-best non-oracle results within each backbone.

Model	Method	Pixel-ready latency (ms) ↓	Speedup ↑	LPIPS ↓	PSNR ↑	Boundary error ↓
minWM	Wait	6815.0	1.00×	0.3159	12.95	0.0215
	Condition Swap	**2498.1**	**2.73**×	0.3155	<u>12.97</u>	0.0215
	Partial Rollback-$d = 1$	2979.3	2.29×	<u>0.3111</u>	12.95	<u>0.0212</u>
	Re-noising (rerun 2)	2556.4	2.67×	0.3160	12.92	0.0214
	Full Rollback	3481.4	1.96×	–	–	–
	CST$_R$	<u>2537.1</u>	<u>2.69</u>×	**0.1214**	**16.30**	**0.0175**
HY-WM1.5	Wait	11711.2	1.00×	0.3682	14.94	0.0379
	Condition Swap	**7052.1**	**1.66**×	0.3446	15.20	<u>0.0368</u>
	Partial Rollback-$d = 1$	7942.8	1.47×	<u>0.1674</u>	<u>18.89</u>	0.0399
	Re-noising	<u>7101.2</u>	<u>1.65</u>×	0.3480	15.08	0.0379
	Full Rollback (oracle)	8706.8	1.35×	–	–	–
	CST$_R$	7124.0	1.64×	**0.0831**	**20.51**	**0.0196**

CST$_T$. Methods within each group share the prompt, initial state, committed history, action update, receipt step, and solver schedule. Stochastic inputs are matched where applicable, with an additional fixed Gaussian draw for re-noising.

We measure reference fidelity using LPIPS and PSNR over the active chunk for CST$_R$ and its editable suffix for CST$_T$. Prefix LPIPS compares the retained region with the matched clamped reference. Boundary error measures the discrepancy between generated and reference frame changes at the start of the editable region. Camera trajectory and HY-WorldPlay benchmark results use arithmetic means under their respective protocols. Blinded human annotations measure action following as the percentage of examples showing the revised action in the editable region and stale frames as the mean number of frames before its first visible response. Pixel-ready latency starts at action receipt and ends when the decoded active chunk is available for immediate intervention methods or when the decoded next chunk is available for waiting. We compute speedup as the ratio of waiting's median latency to the method's median latency. Timing uses an NVIDIA A100 for minWM and an NVIDIA H100 for HY-WM1.5, with comparisons restricted to the same backbone. Appendix A.4 details capture grouping, metric computation, and timing.

4.2 Whole-Chunk Action Updates

We evaluate whole-chunk retargeting with an action update received after $r = 2$ of the $K = 4$ solver evaluations. The revised action is intended to control the entire active chunk. Table 1 compares methods under the matched protocol, with full rollback as the reference. Appendices B.1 and B.3 report sweeps over receipt steps and results for all six directed action transitions.

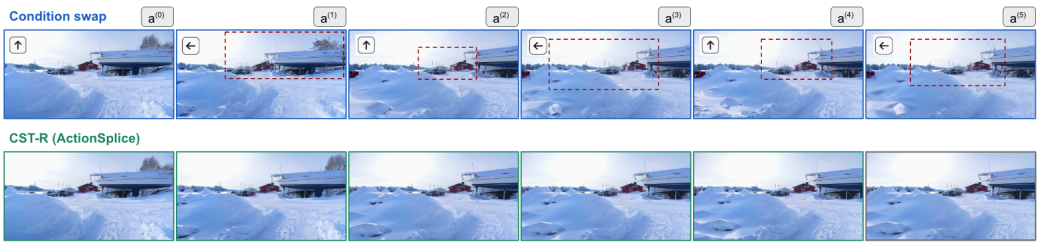

Figure 2: Qualitative HY-WM1.5 rollout with five repeated action updates across six action segments. Each update arrives at $r = 2$ of $K = 4$. Condition swapping (top) retains stale motion after the updates, whereas CST$_R$ (bottom) follows each revised action within the active chunk.

CST_R achieves the lowest LPIPS and boundary error and the highest PSNR among non-oracle methods on both backbones. Compared with condition swapping, it reduces LPIPS by 61.5% on minWM and 75.9% on HY-WM1.5, with PSNR gains of 3.33 and 5.31 dB, respectively. On HY-WM1.5, CST also lowers LPIPS from 0.1674 to 0.0831 relative to partial rollback while reducing latency from 7942.8 to 7124.0 ms. It achieves pixel-ready speedups of $2.69\times$ and $1.64\times$ over waiting on minWM and HY-WM1.5, respectively. These results show lower error against matched rollback without replaying completed solver evaluations.

4.3 Within-Chunk Action Transitions

We evaluate action transitions at boundary m within the active chunk. Positions before m should retain the previous action, while those from m onward should follow the revised action. CST_T restricts its learned correction to this suffix. Table 2 reports aggregate results against rollback with prefix clamping, and Appendix B.2 provides results for each boundary.

Table 2: Action transitions within a chunk at $r = 2$, $K = 4$, over 30 prompts and boundaries $m \in \{1, 2, 3\}$ (90 groups). Fidelity uses matched rollback with prefix clamping. Bold and underlined values mark the best and second best results excluding the oracle for each backbone.

Model	Method	Prefix LPIPS ↓	Suffix LPIPS ↓	Suffix PSNR ↑	Boundary error ↓
minWM	Wait	0.0863	0.2874	13.15	0.0552
	Condition Swap	0.0867	0.2868	13.16	0.0548
	Partial Rollback-$d = 1$	<u>0.0842</u>	<u>0.2773</u>	<u>13.27</u>	<u>0.0542</u>
	Re-noising (rerun 2)	0.0897	0.2889	13.14	0.0549
	Full Rollback (oracle)	–	–	–	–
	CST_T	**0.0740**	**0.1258**	**16.46**	**0.0511**
HY-WM1.5	Wait	0.0715	0.3612	15.06	0.0497
	Condition Swap	0.0763	0.2731	15.53	0.0451
	Partial Rollback-$d = 1$	<u>0.0498</u>	<u>0.0809</u>	<u>21.52</u>	<u>0.0269</u>
	Re-noising (rerun 2)	0.0933	0.2790	15.51	0.0464
	Full Rollback (oracle)	–	–	–	–
	CST_T	**0.0412**	**0.0615**	**25.05**	**0.0215**

CST_T achieves the lowest prefix LPIPS, suffix LPIPS, and boundary error and the highest suffix PSNR among non-oracle methods on both backbones. Compared with condition swapping, it reduces suffix LPIPS by 56.1% on minWM and 77.5% on HY-WM1.5, with suffix PSNR gains of 3.30 and 9.52 dB, respectively. On HY-WM1.5, CST also lowers suffix LPIPS from 0.0809 to 0.0615 relative to partial rollback. Lower prefix LPIPS and boundary error indicate closer agreement with the clamped reference before and across the action boundary.

Table 3 reports blinded human evaluation on HY-WM1.5. CST_R follows the revised action in 96.7% of cases with 1.07 mean stale frames, compared with 83.3% and 8.90 frames for condition swapping. For CST_T, action-following reaches 100.0% with 0.40 mean stale frames, compared with 86.7% and 3.73 frames for condition swapping.

4.4 Comparison with Prior Work

Table 4 reports a secondary evaluation of CST_R under the HY-WorldPlay protocol for 200 prompts used by Lu et al. (2026). Using the reproduced HY-WM1.5 backbone, CST achieves the highest reported PSNR and SSIM and the lowest LPIPS in both comparisons. This evaluation complements the main experiments on in-flight action-update fidelity and latency.

Table 3: Human evaluation of action responsiveness on HY-WM1.5 over 30 matched CST_R groups and 90 matched CST_T groups. Action following measures responses within the editable region; mean stale frames measures the delay to the first visible response.

Model	Method	CST_R		CST_T	
		Action following (%) ↑	Mean stale frames ↓	Action following (%) ↑	Mean stale frames ↓
HY-WM1.5	Wait	20.0	15.37	26.7	7.53
	Condition Swap	83.3	8.90	86.7	3.73
	Partial Rollback-$d = 1$	<u>86.7</u>	<u>3.00</u>	<u>93.3</u>	<u>0.93</u>
	Re-noising	76.7	9.90	86.7	4.40
	Full Rollback (oracle)	100.0	0.00	100.0	0.00
	CST	**96.7**	**1.07**	**100.0**	**0.40**

Table 4: Global video quality under the HY-WorldPlay protocol. CST_R uses reproduced HY-WM1.5. Bold and underlined values mark the best and second-best reported results.

Model	Method	vs. Original			Self-Comparison		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
HY-WorldPlay	Original (HunyuanWorld, 2025)	–	–	–	18.60	0.5678	0.2051
	SVG (Xi et al., 2025)	19.48	0.6028	0.2209	17.75	0.5299	0.2187
	BSA (Meituan LongCat Team et al., 2025)	15.94	0.4639	0.3755	15.44	0.4205	0.3720
	TeaCache (Liu et al., 2025)	20.90	<u>0.6588</u>	0.1892	<u>18.86</u>	0.5743	0.2054
	Light Interaction (Lu et al., 2026)	<u>24.81</u>	0.6500	<u>0.1788</u>	18.85	<u>0.5854</u>	<u>0.1963</u>
	CST_R	**25.66**	**0.6902**	**0.1337**	**20.38**	**0.6267**	**0.1587**

4.5 Camera-Action Following

We evaluate camera-action following with the rotational and translational trajectory errors R_{dist} and T_{dist} from HunyuanWorld (2025). We estimate camera poses with ViPE (Huang et al., 2025) and follow WorldPlay by expressing estimated and target trajectories relative to their first poses and rescaling estimated translations using the furthest target frame. Table 5 reports errors over the active chunk for CST_R and the editable suffix for CST_T. Published results evaluate videos of 61 frames and provide context, although the different evaluation intervals prevent a direct comparison.

Table 5: Camera action following at $r = 2$ over 30 prompts, averaging three boundaries for CST_T. Published results provide context under different evaluation intervals. Lower is better.

Method	R_{dist} ↓	T_{dist} ↓
CameraCtrl (He et al., 2024)	0.037	0.341
VMem (Li et al., 2025)	0.048	0.219
Matrix-Game 2.0 (He et al., 2025)	0.287	0.843
WorldPlay (HunyuanWorld, 2025)	0.031	0.121
CST_R	0.048	0.025
CST_T	0.053	0.054

4.6 Repeated In-Flight Action Updates

We evaluate error accumulation across five action updates and six segments on HY-WM1.5. Each update arrives at $r = 2$ of $K = 4$ solver evaluations. The CST_R corrector is trained on trajectories with three interruptions and evaluated on five-interruption sequences without further training. CST, condition swapping, and sequential full rollback use matched prompts, action schedules, initial states, and stochastic inputs. Figure 3 reports camera-trajectory errors against the requested motion and LPIPS and boundary error against sequential full rollback. Across all five interruptions, CST_R has lower errors than condition swapping. Its rollback-relative LPIPS increases with interruption index, showing growing deviation from the reference. Figure 2 shows a qualitative example of a rollout with six action segments.

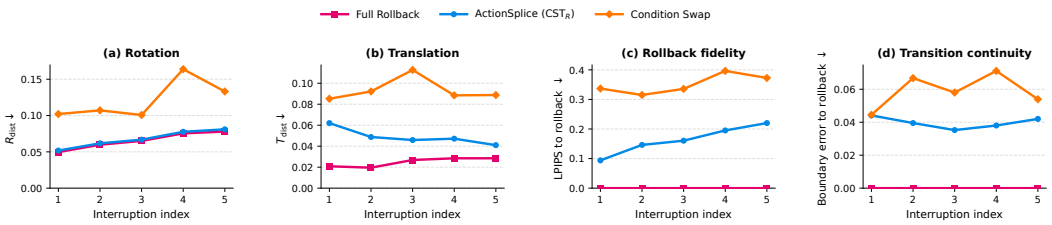

Figure 3: Repeated in-flight updates on HY-WM1.5 at $r = 2$ of $K = 4$. The CST_R corrector is trained on trajectories with three interruptions and tested on five without further training. Curves show mean errors across test prompts. Camera errors are measured against the requested trajectory, while LPIPS and boundary error are measured against sequential full rollback.

4.7 Qualitative Comparisons

Figure 4 shows whole-chunk updates on both backbones. On HY-WM1.5, condition swapping continues forward after a yaw-left request, while CST_R turns within the active chunk. On minWM, waiting delays a yaw-right request until the next chunk, while CST_R responds within the active chunk. In both examples, CST makes the revised action visible without restarting sampling.

Figure 5 shows a within-chunk transition requested at $r = 2$, with a change from yaw left to yaw right at $m = 2$. The comparison rollout follows yaw left throughout the chunk. CST_T retains yaw-left motion in the prefix and follows yaw right in the suffix. Both rollouts continue through the next chunk to show how their trajectories develop.

4.8 Ablations

We ablate matched counterfactual supervision and temporal masking under the full model's evaluation protocol. Removing the mask allows CST_T to modify prefix coordinates and tests the effect of restricting corrections to the suffix. The supervision ablation pairs each source with a target from a different prompt at the same solver step. Table 6 reports action error, rollback-relative fidelity,



Figure 4: Whole-chunk action-update examples. Top pair shows a forward to yaw-left request on HY-WM1.5, where condition swapping continues forward motion. Bottom pair shows yaw-left to yaw-right request on minWM, where waiting delays the update until next chunk. CST_R responds within the active chunk in both examples. Red and green borders mark stale and responsive frames.

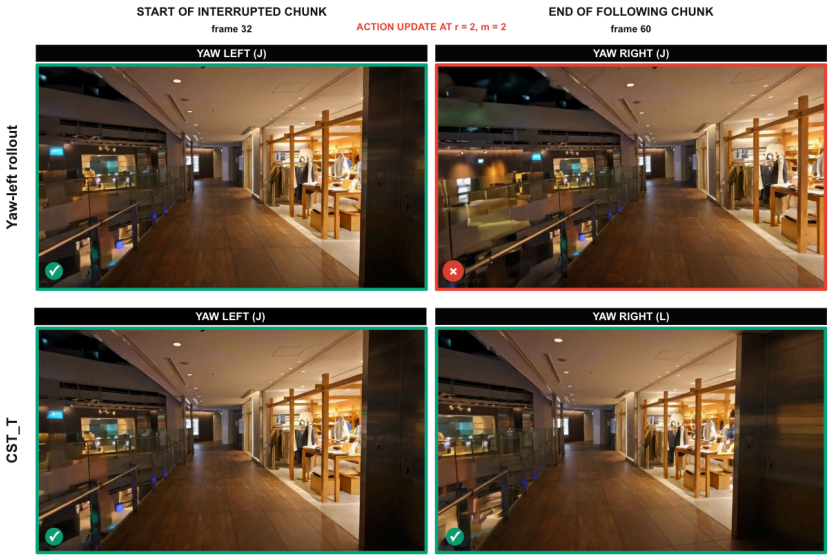

Figure 5: Within-chunk action transition requested at $r = 2$ with boundary $m = 2$. Top rollout continues yaw left, while CST_T follows the requested change from yaw left to yaw right. Columns show the start of the interrupted chunk and the end of the following chunk. Green and red borders indicate agreement and disagreement with the requested action, respectively.

boundary error, and prefix LPIPS. Action error averages the sum of rotation and translation errors between estimated and requested relative camera motions, normalized by their respective command magnitudes. Appendix A.5 provides the formula and ablation settings.

Table 6: ActionSplice ablations on HY-WM1.5. RB-LPIPS compares the active chunk with matched full rollback for CST_R and the editable suffix with matched prefix-clamped rollback for CST_T. Prefix LPIPS uses the clamped reference. Lower is better.

Configuration	CST_R			CST_T			
	Action error ↓	RB-LPIPS ↓	Boundary error ↓	Action error ↓	RB-LPIPS ↓	Boundary error ↓	Prefix LPIPS ↓
Full model	**0.6192**	**0.0831**	**0.0196**	**0.3283**	**0.0615**	**0.0215**	**0.0412**
Without hard mask	–	–	–	0.4092	0.0787	0.0240	0.0604
Unmatched supervision	4.6889	0.7797	0.0435	3.0415	0.7719	0.0349	0.4209

The full model achieves the lowest error on every reported metric. Without the hard mask, CST_T prefix LPIPS rises from 0.0412 to 0.0604 and boundary error from 0.0215 to 0.0240. Unmatched supervision worsens every metric, supporting the use of matched source-target pairs.

5 Conclusion

We introduced ActionSplice for incorporating action changes into an active chunk during sampling. A learned corrector moves the interrupted state representation toward its matched counterfactual. CST_R retargets the active chunk, while CST_T restricts the correction to a temporal suffix. Across minWM and HY-WM1.5, CST improves fidelity to matched rollback relative to condition swapping. CST_R also reduces pixel-ready latency relative to waiting and maintains lower errors than condition swapping across five repeated updates. CST enables sampling to resume under revised actions without repeating completed solver evaluations.

Reproducibility Statement

We provide implementation and evaluation details throughout the paper and appendix. Section 3 describes Counterfactual State Transport, including matched counterfactual targets, state transport, and corrector training. The experimental protocol, baselines, metrics, and data splits are described in Section 4.1, with additional implementation and optimization details in Appendices A.1 and A.3. Appendix A.4 specifies the evaluation protocol, including matched-group construction, metric computation, and timing. Additional ablations and analyses of receipt-step sensitivity, temporal boundaries, and directed action transitions are provided in Appendices A.5–B.3.

AI Use Statement

Generative AI tools were used to assist with the clarity and presentation of the manuscript, as well as with limited code debugging. All research decisions were made by the authors, who independently verified the implementations, experiments, analyses, and reported results. The authors reviewed all AI-assisted content and take responsibility for the final manuscript and its claims.

References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Xiangbo Gao, Siyuan Yang, Ping He, Mingyang Wu, Yuheng Wu, Yushen Zuo, Jiongze Yu, Ryan Cui, Hongyuan Hua, Devin Ma, et al. Visko orbis 1.0: A live model for real-time interactive long video generation. *arXiv preprint arXiv:2607.26694*, 2026.

Peyman Gholami and Robert Xiao. Streamlining image editing with layered diffusion brushes. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17368–17378. IEEE, 2025.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.

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

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

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

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

Seonsoo Kim, Seongil Hong, and Jun-Gill Kang. Diffusion reroll: Revisable denoising for robotic sequential prediction. *arXiv preprint arXiv:2607.19919*, 2026.

Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. Vmem: Consistent interactive video scene generation with surfel-indexed view memory. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 25690–25699. IEEE, 2025.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363. IEEE, 2025.

Hao Liu, Ye Huang, Chenghuan Huang, Zhenyi Zheng, Jiangsu Du, Ziyang Ma, Jing Lyu, and Yutong Lu. Beyond few-step inference: Accelerating video diffusion transformer model serving with inter-request caching reuse. *arXiv preprint arXiv:2604.04451*, 2026.

Jiacheng Lu, Haoyi Zhu, Sipei Yi, Enze Xie, Yu Li, and Cheng Zhuo. Light interaction: Training-free inference acceleration for interactive video world models. *arXiv preprint arXiv:2605.31158*, 2026.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. Longcat-video technical report. *arXiv preprint arXiv:2510.22200*, 2025. URL `https://arxiv.org/abs/2510.22200`.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15886–15896. IEEE, 2023.

Shaohao Rui, Xiaofeng Mao, Zhanyu Zhang, Peijia Lin, Yansong Zhu, Yibo Zhang, Haibin Wan, Zhangrui Zhao, and Weijie Ma. Biwm: Advancing open-source interactive video world models with bidirectional autoregression. *arXiv preprint arXiv:2606.10135*, 2026.

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

Pardis Taghavi, Reza Langari, and Gaurav Pandey. Partition the support, reconstruct the residual: Training-free sparse attention for video generation and world models. *arXiv preprint arXiv:2608.18484*, 2026.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, volume 2025, pp. 73754–73776, 2025.

Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22532–22541. IEEE, 2023.

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

Mingqiang Wu, Weilun Feng, Zhefeng Zhang, Haotong Qin, Yuqi Li, Guoxin Fan, Xiaokun Liu, Zhulin An, Libo Huang, Yongjun Xu, et al. Echo-forcing: A scene memory framework for interactive long video generation. *arXiv preprint arXiv:2605.16003*, 2026a.

Yuheng Wu, Xiangbo Gao, Tianhao Chen, Xinghao Chen, Qing Yin, Zhengzhong Tu, and Dongman Lee. Delta forcing: Trust region steering for interactive autoregressive video generation. *arXiv preprint arXiv:2605.14382*, 2026b.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025.

Yang Yang, Tianyi Zhang, Wei Huang, Jinwei Chen, Boxi Wu, Xiaofei He, Deng Cai, Bo Li, and Peng-Tao Jiang. Anchor forcing: Anchor memory and tri-region rope for interactive streaming video diffusion. *arXiv preprint arXiv:2603.13405*, 2026.

Yixiao Zeng, Jianlei Zheng, Chaoda Zheng, Shijia Chen, Mingdian Liu, Tongping Liu, Tengwei Luo, Yu Zhang, Boyang Wang, Linkun Xu, et al. X-cache: Cross-chunk block caching for few-step autoregressive world models inference. *arXiv preprint arXiv:2604.20289*, 2026.

Min Zhao, Hongzhou Zhu, Bokai Yan, Zihan Zhou, Yimin Chen, Wenqiang Sun, Kaiwen Zheng, Guande He, Xiao Yang, Chongxuan Li, et al. minwm: A full-stack open-source framework for real-time interactive video world models. *arXiv preprint arXiv:2605.30263*, 2026a.

Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal forcing++: Scalable few-step autoregressive diffusion distillation for real-time interactive video generation. *arXiv preprint arXiv:2605.15141*, 2026b.

Weisen Zhao, Lam Nguyen, Zhicong Lu, and Yuzhang Shang. C^{3}ache: Accelerating world action models with cross inference chunk cache. *arXiv preprint arXiv:2606.08962*, 2026c.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *International Conference on Learning Representations*, volume 2025, pp. 3296–3319, 2025.

Kaiwen Zheng, Guande He, Min Zhao, Jintao Zhang, Huayu Chen, Jianfei Chen, Chen-Hsuan Lin, Ming-Yu Liu, Jun Zhu, and Qianli Ma. Causal-rcm: A unified teacher-forcing and self-forcing open recipe for autoregressive diffusion distillation in streaming video generation and interactive world models. *arXiv preprint arXiv:2606.25473*, 2026.

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

Supplementary Material

A Implementation and Training Details

A.1 Backbone-Specific Transport and Resumption

minWM receives camera view matrices and intrinsics, while HY-WM1.5 also receives discrete action labels. For CST_T, the conditioning switches action at latent position m. Counterfactual target construction first computes the source trajectory, then clamps the target prefix using source quantities from the same solver step. The target is captured at receipt step r. This source trajectory is not used during inference. During continuation, minWM restores the prefix cached at receipt, while HY-WM1.5 continues without repeated prefix replacement.

The corrector receives the interrupted and initial states, the preceding committed chunk, old and revised action conditioning, and the receipt step. HY-WM1.5 continuation is deterministic and requires no noise trace. For CST_T, the temporal mask is provided as an input channel and also restricts the predicted residual. CST_R additionally conditions on the interruption index, while CST_T does not.

Each corrector uses six FiLM residual blocks in a 3D encoder-decoder. Two spatial downsampling stages use channel widths 128, 256, and 512, followed by interpolation and skip additions. Each

block combines a $1\times3\times3$ spatial convolution with a $3\times1\times1$ temporal convolution, GroupNorm, and SiLU. Camera conditioning contains 45 features per latent position from old, revised, and relative poses and normalized intrinsics. A two-layer MLP embeds each position. We pool the mean, first-position, and last-position embeddings and combine them with receipt-step conditioning and, for CST_R, an interruption-index embedding. The conditioning width is 256. minWM and HY-WM1.5 use 16 and 32 latent channels, respectively.

HY-WM1.5 rebuilds its context cache from committed frames before continuation, while minWM reuses its existing caches.

A.2 Dataset Construction

Each prompt has one fixed seed. Captures follow the prompt-disjoint split and action transitions described in the main text. Each capture stores source and target solver traces from which training pairs are extracted at receipt steps $r \in \{1, 2, 3\}$.

For CST_R, each prompt produces one recurrent trajectory with three interruption events and four action segments. Each action persists for three or four chunks, which produces trajectories of 48 to 64 temporal latent positions. The three events yield 540 captures in total, with 360 for training, 90 for validation, and 90 for testing. For CST_T, each prompt produces one recurrent capture for each boundary $m \in \{1, 2, 3\}$. This gives the same 360/90/90 capture split. The six action transitions and three temporal boundaries are balanced within each partition. Capture counts refer to interruption events, not to the number of solver-step pairs extracted from their saved traces. Later events use committed chunks generated after earlier corrections. Source and target branches share this history at each event, and training does not backpropagate through capture generation.

A.3 Training Objective and Optimization

We train each corrector for 4,000 steps with AdamW, batch size 1, weight decay 10^{-4}, and gradient clipping at 1.0. The learning rate follows a cosine schedule from 10^{-5} to 10^{-6}. Each backbone and variant has a separate corrector trained only with representation-matching NMSE while the world model and sampler remain frozen.

We average the normalized loss from Section 3.4 across examples. For CST_R, the loss covers the full state. For CST_T, it covers only the editable suffix. This loss mask is distinct from the mask applied to the predicted correction.

We maintain an exponential moving average of the corrector weights with decay 0.999 and evaluate it every 200 steps on the validation set. We select the EMA checkpoint with the lowest validation transport NMSE and use it for test evaluation.

A.4 Evaluation Protocol

Comparison groups. We use the matched groups defined in the main text and aggregate each backbone and variant separately. For CST_R, we average the three interruption-level measurements within each prompt, producing 30 prompt groups from 90 test captures. Receipt-step sweeps vary r within the same trajectory. Each CST_T prompt-boundary pair is one group, giving 90 groups; the three boundaries for a prompt are not independent scenes.

Decoded evaluation intervals. For a rollout with L latent positions and F decoded frames, let j be the active chunk's first latent index. Decoded indices are

$$g(u) = \mathrm{round}\left(u\frac{F-1}{L-1}\right), \qquad b = g(j+m), \qquad e = \min\{F, g(j+T)\}. \tag{4}$$

Rounding uses the nearest integer. The active chunk, editable region, and prefix occupy $[g(j), e)$, $[b, e)$, and $[g(j), b)$, respectively. The boundary m indexes latent positions.

Reference fidelity. Let x_t and $x_t^\star$ be generated and matched rollback frames with RGB values in $[0, 1]$. The reference follows Section 3.2, including prefix clamping for CST_T. For the editable

decoded interval $\mathcal{E} = [b, e)$, we average framewise AlexNet LPIPS and compute PSNR from the region's mean squared error,

$$\mathrm{MSE}_{\mathcal{E}} = \frac{1}{|\mathcal{E}|D} \sum_{t \in \mathcal{E}} \|x_t - x_t^\star\|_2^2, \qquad \mathrm{PSNR}_{\mathcal{E}} = 10 \log_{10} \frac{1}{\mathrm{MSE}_{\mathcal{E}}}, \tag{5}$$

where D is the number of scalar RGB values per frame. Prefix LPIPS averages framewise LPIPS over $[g(j), b)$ against the clamped reference. It measures reference agreement and does not establish exact preservation of the original decoded prefix.

Boundary error and aggregation. Boundary error compares generated and reference frame changes at the first editable decoded frame,

$$E_{\mathrm{boundary}} = \frac{1}{D} \left\| (x_b - x_{b-1}) - (x_b^\star - x_{b-1}^\star) \right\|_1 . \tag{6}$$

The preceding frame belongs to committed history for CST_R and to the prefix for CST_T. This metric measures agreement with the reference transition, rather than smoothness alone. We compute one score per group and method, then report the median across groups for LPIPS, PSNR, and boundary error. Camera-trajectory and HY-WorldPlay benchmark results use means. The overall median is computed from individual groups, not from subgroup medians.

Camera trajectories. We align estimated and requested camera-to-world poses at common frame indices and express each trajectory relative to its first pose. Let $(\widehat{R}_i, \widehat{t}_i)$ and (R_i, t_i) denote these aligned poses. We normalize estimated translation by

$$k = \arg\max_i \|t_i\|_2, \qquad \alpha = \frac{\|t_k\|_2}{\|\widehat{t}_k\|_2}. \tag{7}$$

Alignment and scale estimation use the full aligned trajectory before selecting the scored pose indices $\mathcal{P}$. Following HunyuanWorld (2025), R_{dist} averages geodesic rotation error in radians and T_{dist} averages squared Euclidean translation error between $\alpha\widehat{t}_i$ and t_i. We then average these per-example scores. The scored interval contains four poses for CST_R and $4-m$ suffix poses for CST_T, unlike the published 61-frame evaluations.

Timing and computation. The $K - r$ remaining solver evaluations exclude corrector execution, state reconstruction, context-cache processing, and decoding. Cache rebuilding requires additional transformer work, so this count does not measure total transformer calls or runtime. Reference construction is excluded from CST timing.

A.5 Ablation Details

The ablations use separately trained HY-WM1.5 correctors and the same matched evaluation protocol as the full model. Unmatched supervision assigns each source capture a target from a different prompt through a fixed, seeded permutation within its data partition. Source inputs are retained, and the target is taken at the same solver step. This breaks the counterfactual pairing without mixing training and validation prompts.

The mask ablation disables both the mask input channel and multiplication of the predicted residual by the temporal mask during training and inference. The suffix mask still selects the elements used in the matching loss. Prefix coordinates can therefore change without direct prefix supervision. This ablation tests the combined effect of mask conditioning and output restriction. It does not isolate output masking alone.

Action error measures disagreement between estimated and requested relative camera motions. After the alignment above, let $\Delta\widehat{R}_i$ and ΔR_i be successive relative rotations, and $\Delta\widehat{t}_i$ and Δt_i the corresponding translations in the preceding camera frame. Let $\theta(Q)$ denote the geodesic angle of rotation Q in radians. For increments entering the scored pose interval $\mathcal{P}$, we compute

$$E_{\mathrm{action}} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \left[\frac{\theta(\Delta\widehat{R}_i \Delta R_i^\top)}{\pi/60} + \frac{\|\alpha\Delta\widehat{t}_i - \Delta t_i\|_2}{0.08} \right]. \tag{8}$$

The normalizers are the command magnitudes of 3 degrees and 0.08 scene units. We include the increment from the preceding pose into the first scored pose, giving four increments for CST_R and $4 - m$ for CST_T. Table 6 reports the median per-example action error. Its fidelity and boundary metrics use the references and aggregation defined above.

B Additional Results and Human Evaluation

B.1 Receipt-Step Sensitivity

Table 7: Receipt-step sweep for CST_R on HY-WM1.5 with $K = 4$. Metrics follow Appendix A.4.

Receipt step r	Remaining solver evaluations	Pixel-ready latency (ms) ↓	LPIPS ↓	PSNR ↑	Boundary error ↓	R_{dist} ↓	T_{dist} ↓
1	3	7578.8	0.0643	23.13	0.0204	0.108	0.024
2	2	7124.0	0.0831	20.51	0.0196	0.048	0.025
3	1	6026.6	0.2694	18.39	0.0246	0.180	0.064

Later receipt reduces latency, but $r = 3$ has higher LPIPS and boundary error and lower PSNR than $r = 1$ or $r = 2$.

B.2 Temporal-Boundary Sensitivity

Table 8: Temporal-boundary sweep for CST_T on HY-WM1.5 at $r = 2$. Scored intervals, references, and metrics follow Appendix A.4.

Boundary m	Prefix LPIPS ↓	Suffix LPIPS ↓	Suffix PSNR ↑	Boundary error ↓	R_{dist} ↓	T_{dist} ↓
1	0.0403	0.0701	24.74	0.0189	0.062	0.047
2	0.0443	0.0597	25.53	0.0208	0.049	0.093
3	0.0395	0.0578	24.69	0.0230	0.049	0.023

The scored suffix shortens as m increases, so values across boundaries refer to different temporal regions.

B.3 Directed Action-Transition Breakdown

Table 9: Action-transition breakdown for CST_R on HY-WM1.5 at $r = 2$. Each row reports mean camera errors and median LPIPS under Appendix A.4. The final row pools all 30 examples.

Action transition	R_{dist} ↓	T_{dist} ↓	LPIPS to rollback ↓
Forward → Backward	0.002	0.028	0.0479
Forward → Yaw-left	0.078	0.031	0.1285
Backward → Forward	0.003	0.043	0.0593
Backward → Yaw-left	0.054	0.003	0.1042
Yaw-left → Forward	0.074	0.028	0.1434
Yaw-left → Backward	0.079	0.019	0.2654
All transitions	0.048	0.025	0.0831

Performance varies by transition, with the highest rollback-relative LPIPS for yaw-left to backward.

B.4 HUMAN ANNOTATION INTERFACE

Figure 6 shows the blinded annotation interface. Randomized sample identifiers hide method identity. For each action segment, annotators record whether the requested action is visible in the editable region and count decoded frames from the start of that region to the first visible response. The scored region is the active chunk for CST_R and its suffix for CST_T. We report action following as a percentage and stale frames as a mean.

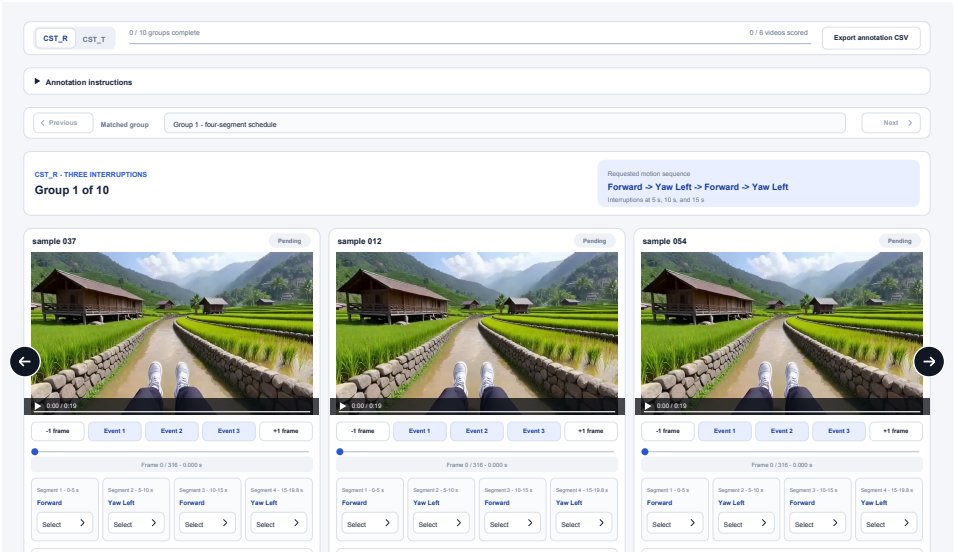

Figure 6: Blinded action-response annotation interface. Randomized sample identifiers conceal method identity, and segment controls support frame-level inspection around each action update.