LONG-HORIZON VIDEO GENERATION WITH SLOT MEMORY ADDRESSED BY REGION TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory for long-horizon video generation must satisfy two competing requirements: it must remain bounded so generation can continue indefinitely, while still recovering what was observed at a place when the camera returns, regardless of how long ago it was seen. Reconstruction-based memories provide spatial recall by caching and reprojecting explicit geometry, but require growing storage and depend on estimated 3D structure. Recurrent memories keep a constant-size state, but retention is governed by temporal recurrence, causing old observations to fade with time. We introduce SMART, a slot memory addressed by region tokens, which combines bounded storage with spatially addressed recall. SMART stores content in a constant-size slot recurrence whose writes are addressed by the observing camera and whose reads are queried by the rendering camera, without constructing an explicit 3D representation or maintaining a growing cache. Consequently, recall is conditioned on inter-view geometry rather than on recency. Each read additionally provides a confidence estimate that conditions a chunk-causal diffusion transformer, allowing it to selectively trust retrieved content while relying on its generative prior elsewhere. On long trajectories in Memory Maze and Minecraft, SMART maintains high-fidelity recall far beyond its training horizon while using constant memory and per-step cost.

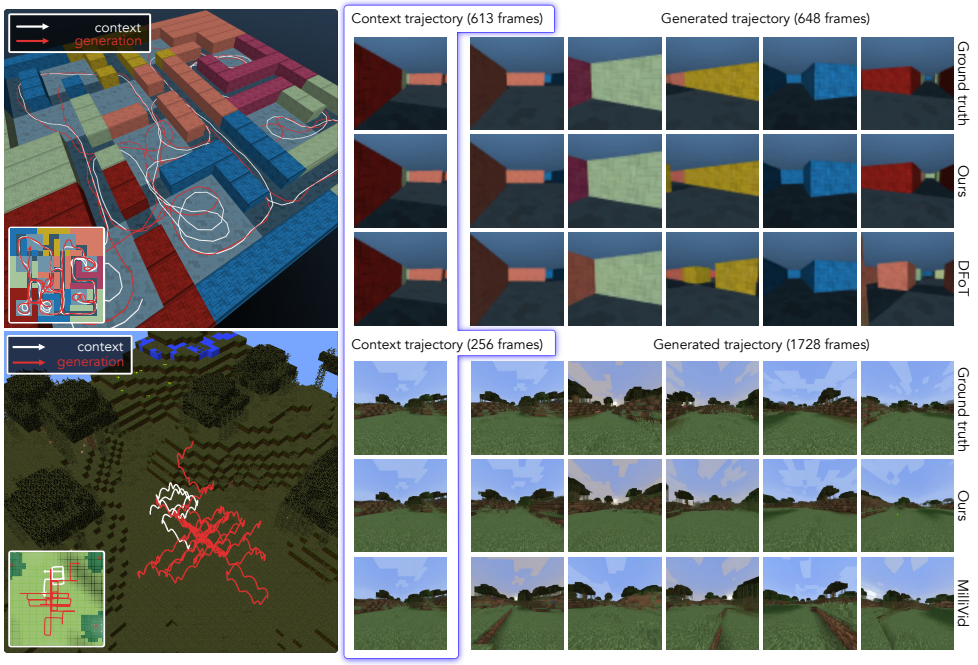


Figure 1: **Long-horizon video generation.** *Top:* on *Memory Maze*, SMART and DFoT (Song et al., 2025) observe a forward trajectory and generate the return leg, revisiting places seen long before. *Bottom:* on *Minecraft*, SMART and MilliVid (Chandratreya et al., 2026) generate a continuation trajectory from 256 context frames written into the memory. SMART retains observed content more faithfully than the baselines at constant per-step cost.

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

Video generative models have improved dramatically in recent years. Large scale training and autoregressive generation have enabled models to produce increasingly long videos while following camera and action controls with high fidelity. However, even compressed latent videos consist of thousands of tokens, making it costly to train and condition on long sequences. Moreover, a generative model must implicitly learn complex long-range dependencies, such as those imposed by multi-view consistency. Even large-scale autoregressive models (Bahmani et al., 2026; Chen et al., 2026b; 2024) struggle with this: when an agent walks down a corridor and subsequently retraces its path, the environment may no longer match its initial state. Remembering scene geometry over long horizons therefore remains difficult even at scale. Conditioning on a persistent representation of previously observed content can alleviate both problems. Rather than requiring the model to retain the full history and reconstruct cross-view relationships implicitly, such a representation can summarize past observations and provide information relevant to the current viewpoint. The central question thus becomes, how this representation should be structured.

Two families of methods have emerged to equip video generation with memory, which differ in terms of how memory is addressed. *Explicit* memory directly instantiates the underlying 3d geometry in the form of Gaussians, point clouds or a latent 3D map, and reprojects it to the target camera to condition the generator (Bahmani et al., 2026; Shen et al., 2026; Yu et al., 2026; Wei et al., 2026; Zhao et al., 2026; Wang et al., 2026). The memory is addressed by a spatial location: content can be retrieved according to where it was observed. Explicit methods, therefore, enforce some level of long-term consistency by construction. However, the required storage generally grows with the size of the explored scene unless information is explicitly pruned, and the resulting rendered conditioning provides the generator only with limited information about which reconstructed regions are reliable. *Implicit* memory, in the form of linear-time recurrent sequence models, stores the memory within the neural architecture (Po et al., 2025; Savov et al., 2026): in language models, state-space approaches (Gu et al., 2021; Gu & Dao, 2023; Dao & Gu, 2024), gated linear attention and slot recurrences (Zhang et al., 2024; Du et al., 2026; Afzal et al., 2026) all compress an unbounded history into a constant-size state, evaluated on exactly the retrieval regime that concerns us. However, such memory schemes lack an explicit spatial addresses: information is accumulated through temporal recurrence rather than stored according to where it was observed. Moreover, every frame updates the entire state and decays older information, so what is retained inherently depends on recency rather than spatial location. Old observations beyond the training horizon inevitably fade out.

We propose a conditioning mechanism that combines the complementary strengths of both approaches: the bounded, learned representation of implicit memory and the spatial addressing of explicit geometry. We introduce SMART (**s**lot **m**emory **a**ddressed by **r**egion **t**okens), which couples a fixed-capacity implicit memory, in which camera pose determines where observations are written and from where they are retrieved, with a video diffusion model conditioned on what that memory returns. Rather than asking the model to recover an observation from a particular (past) time step, a query decodes the memory into a coarse representation of the target view together with a confidence estimate. This provides the generative model with a full-frame conditioning signal, preserving the convenient interface of explicit reconstruction-based methods; but without the requirement to keep a full 3D scene reconstruction in memory. Our contributions are:

- A fixed-capacity slot-based implicit scene memory that addresses observations by where rather than when they were seen, enabling geometry-guided writing, retrieval, and novel-view decoding.
- A memory-conditioned video diffusion model that takes the retrieved view as a coarse draft and refines it into video, falling back on its generative prior where memory is uninformative.

2 RELATED WORK

Video diffusion and controllable generation. Diffusion models have become a strong foundation for video generation, with large-scale architectures such as Wan (Wan et al., 2025) providing high-quality spatiotemporal synthesis. Camera-controlled generation further conditions on viewing geometry, using Plücker-ray representations (He et al., 2025), geometry-aware attention (Miyato et al., 2024), or relative-camera positional encodings (Li et al., 2025; 2026). These mechanisms condition the current

step on pose, but leave open what the model retains from earlier steps.

Temporal context for video generation. Autoregressive and causal diffusion methods extend video generation beyond a fixed temporal window, including Diffusion Forcing (Chen et al., 2024) and history-guided generation (Song et al., 2025). FramePack (Zhang et al., 2025) compresses previous frames into a bounded context, while MilliVid (Chandratreya et al., 2026) uses hierarchical latents and coarse-to-fine rollout to preserve long-range structure. These approaches reduce the cost of long-horizon generation, but recall stays indexed by recency: a region observed long ago is compressed or dropped regardless of whether the camera is about to return to it.

Explicit spatial memory. Several methods equip video generators with explicit scene representations. Gen3C (Ren et al., 2025), Lyra and Lyra 2.0 (Bahmani et al., 2026; Shen et al., 2026), MosaicMem (Yu et al., 2026), and geometry-aware world models (Wei et al., 2026; Xiao et al., 2026) use reconstructed geometry, cached views, or spatial representations to retrieve information relevant to the current camera. Closely related ideas also appear in reconstruction systems with persistent or online state (Wang* et al., 2025; Wang & Agapito, 2025; Chen et al., 2026a; Zhuo et al., 2026). Such spatial memories naturally support revisits, but tie recall to an explicit reconstruction or cache whose representation and storage must be maintained as the explored scene grows. We instead retain spatially relevant information in a fixed-capacity learned state.

Implicit and recurrent memory. State-space models (Gu & Dao, 2023; Dao & Gu, 2024) compress arbitrarily long sequences into recurrent states at linear cost and have been adapted to long-context video (Po et al., 2025). Gated-slot attention (Zhang et al., 2024) and sparse-slot memories such as Raven (Afzal et al., 2026) partition the state into selectively updated slots, so information persists when unrelated inputs arrive. Learned states can also support geometric queries: CUT3R (Wang* et al., 2025) predicts pointmaps from from its recurrent state at queried viewpoints. GIM-World (Wei et al., 2026) compresses past observations into a fixed-size representation with camera-queryable geometry supervision. Yet, CUT3R requires explicit 3D supervision and GIM-World does not maintain a low-cost recurrent memory.

SMART focuses on geometric addressing within the recurrence itself: camera pose governs autoregressive memory updates and readouts. At generation time, a target-camera query decodes a coarse full-frame draft to condition a chunk-causal video diffusion model only on the relevant information.

3 METHOD

Our aim is the following: given a viewpoint trajectory through a scene, generate the video along it such that whenever a viewpoint re-observes a part of the scene, the generated content agrees with the history.

Overview. As shown in Fig. 2, SMART achieves this with two components. A recurrent slot memory stores content under pose-derived keys and, when queried with a target camera, returns a coarse draft of that view together with a confidence estimate in latent space. A chunk-causal diffusion model sharpens the draft into video, trusting each cell in proportion to the memory's confidence. Generated chunks are written back into the memory, so rollouts of arbitrary lengths run at a constant memory size. A detailed introduction to the individual components as well as the preliminaries follow below.

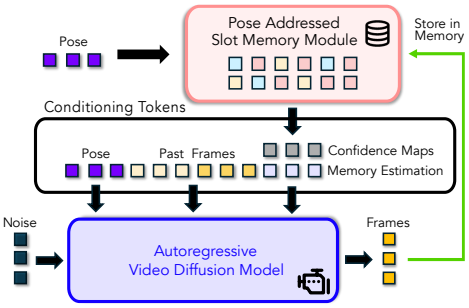


Figure 2: **SMART inference overview.**

3.1 PRELIMINARIES

State-space models. A state-space model processes a sequence through a recurrent state $\boldsymbol{h}_t$ of fixed size, $\boldsymbol{h}_t = \boldsymbol{A}_t\boldsymbol{h}_{t-1} + \boldsymbol{B}_t\boldsymbol{u}_t$ with output $\boldsymbol{y}_t = \boldsymbol{C}_t\boldsymbol{h}_t$, so that arbitrarily long histories are compressed at $O(1)$ cost per token. Structured SSMs (Gu et al., 2021) parameterize $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ for stable long-range propagation, and selective variants (Gu & Dao, 2023; Dao & Gu, 2024) make them input-dependent, letting the model choose per token what to write and what to forget. The state is nonetheless a decaying summary ordered by time: what can be recovered from $\boldsymbol{h}_t$ about an input $\boldsymbol{u}_{t-n}$ diminishes with n. This is the property our memory design must overcome, and the slot-structured recurrence below is the halfway point between such dense states and full attention.

State-space models with Gated Slot Memory. Gated linear attention (Yang et al., 2023; Dao & Gu, 2024) maintains a recurrent key–value state with a learned decay, and slot-structured variants (Zhang et al., 2024) replace the single dense state with M discrete slots per head, each holding a key–value pair updated by an exponential moving average $\boldsymbol{H}_m^{\{k/v\}} \leftarrow \exp(g)\, \boldsymbol{H}_m^{\{k/v\}} + s\,\{\boldsymbol{k}/\boldsymbol{v}\}$ where $g \le 0$ is a per-head log-decay and s a write gate. A read with query $\boldsymbol{q}$ forms $\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{q} \cdot \boldsymbol{H}^k)$ over the M slots and returns $\boldsymbol{o} = \boldsymbol{\alpha} \cdot \boldsymbol{H}^v$. Unlike a dense SSM state, a slot is an addressable unit: what a read returns is decided by the match between the query and the keys under which content was written, not by when the content arrived. We adopt this recurrence as the substrate of our memory, and describe our modifications in Section 3.2.

Latent Video Diffusion. Diffusion models (Ho et al., 2020; Song et al., 2021) generate by learning to invert a gradual noising of the data: a sample $\boldsymbol{x}_0$ is mixed with noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to $\boldsymbol{x}_\tau$ at level $\tau \in [0, 1]$, and a network is trained to recover the clean sample, for instance by predicting the noise, $\mathcal{L}_{\text{diff}} = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\epsilon}, \tau}\left[\, \|\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_\tau, \tau, \mathcal{C}) - \boldsymbol{\epsilon}\|_2^2 \,\right]$, with $\mathcal{C}$ the conditioning. Generation runs the reverse process from pure noise, and flow matching (Lipman et al., 2022) is an equivalent velocity parameterization of the same family. We operate in the latent space of a frozen video VAE (Kingma & Welling, 2014) that downsamples spatially and, depending on the autoencoder, temporally.

3.2 Spatially-Structured Memory Design

The memory is a bank of M slots per head across N_h heads, written as posed latent frames arrive and read by presenting a target camera. Its design enforces one separation: everything that decides *where* content goes (keys, queries, and routing) is computed from pose alone, while *what* is stored, and how strongly it overwrites, is computed from content. Pose enters as data on the addressing side, as an absolute code matched between writer and reader, and as an operator on the content side, where the relative camera between writer and reader transports stored values and warps the decoder's attention. Fig. 5 summarizes the write and read paths. Details beyond this section are given in Section A.3.

Setup and tokenization. A clip is N posed video frames $\mathcal{X} = \{\boldsymbol{x}_i\}$ with cameras $\pi_i = (\boldsymbol{E}_i, \boldsymbol{K}_i)$, the extrinsics $\boldsymbol{E}_i \in SE(3)$ rebased so that $\boldsymbol{E}_0 = \boldsymbol{I}$ and normalized in scale. A frozen video VAE encodes the clip to T latent frames $\boldsymbol{z}_t \in \mathbb{R}^{C \times H_z \times W_z}$, and everything downstream runs at the latent rate. Each latent frame is patch-embedded into P tokens and compressed by a tiled tokenizer (Figure 3): the frame is divided into J fixed image tiles, and write token j cross-attends on the patches of tile j, so $\boldsymbol{c}_{t,j}$ summarizes a fixed region of the frame before the per-head value projection. No spatial or temporal embedding is added to a write token, and temporal order is represented only by the recurrence itself.

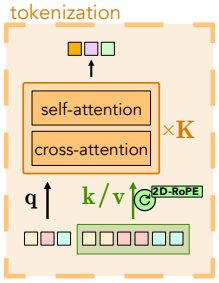


Figure 3: **Tiled tokenization.** Write token j queries only the patches of tile j.

Pose addressing. The address of a region is computed based on viewing geometry alone, i.e., content never enters it. For each cell of a coarse grid we form a footprint ray field given by the Plücker coordinates of the ray,

$$\boldsymbol{r}_i = \Big[\, \boldsymbol{p}_i \times \boldsymbol{d}_{i,uv} \;;\; \boldsymbol{d}_{i,uv} \,\Big]_{u,v}, \tag{1}$$

with $\boldsymbol{p}_i$ the camera center and $\boldsymbol{d}_{i,uv}$ the unit ray through cell (u, v). Camera centers are expressed in units of a scene-scale normalizer σ, keeping the moment term in a stable input range across scenes. The raw geometric field is what lets displaced and co-observing queries land near the writes that cover them, and we verify this in Section 5. A convolutional encoder $\mathcal{E}_{\text{cam}}$ maps the field to an address grid $\boldsymbol{A}_t \in \mathbb{R}^{D \times H_z \times W_z}$, and a second tiled tokenizer ϕ_{cam} reduces it to one address per token and head,

$$[\boldsymbol{a}_{h,t,1}, \ldots, \boldsymbol{a}_{h,t,J}] = \phi_{\text{cam}}(\boldsymbol{A}_t), \qquad \boldsymbol{a}_{h,t,j} \in \mathbb{R}^{d_a}, \tag{2}$$

Thus, region token $\boldsymbol{a}_{h,t,j}$ carries address information for the same image tile as content token, $\boldsymbol{c}_{t,j}$. Write keys and read queries are per-head identity-initialized linear functions of the address alone, $\boldsymbol{k}^h_{t,j} = \boldsymbol{W}^h_k\, \boldsymbol{a}^h_{t,j}$, $\boldsymbol{q}^h_{t^*,j} = \boldsymbol{W}^h_q\, \boldsymbol{a}^h_{t^*,j}$, so the match between a query and a key depends only on the viewing geometry of the two regions, never on their content or on when they were written. When the VAE compresses time, each latent frame inherits f cameras, and $\mathcal{E}_{\text{cam}}$ folds them causally at the VAE's own rate, so the address of latent frame t depends on no later camera.

Routed writing. Every token write touches a single slot and leaves the rest unchanged. Each slot learns, through a fixed embedding, a region of pose space it is responsible for (the rows of a linear router $\boldsymbol{W}_r$), and an incoming address activates the slot per head whose region matches it best, with strength $r_m \in [0, 1)$ given by the match (Figure 4). Scoring against fixed embeddings rather than against the slots' current contents keeps each slot stable over time and improves long-horizon recall, irrespective of what was written in between. Before writing, the value is canonicalized with the inverse writing camera projection, $\tilde{\boldsymbol{v}}$, so content observed from different viewpoints accumulates in a shared frame and can be re-posed to the target camera when read (Miyato et al., 2024; Li et al., 2025). A selected slot then blends new content into what it holds with a content-derived decay $\delta \leq 0$,

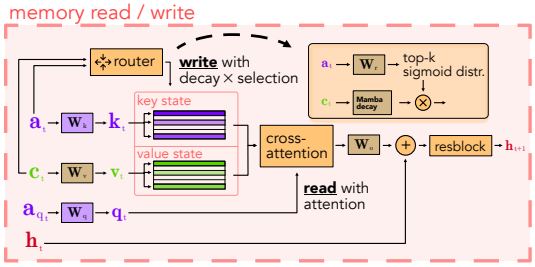


Figure 4: **Slot read and write.** Address routes the write; Read is a cross-attention over slot states.

$$g_m = \delta\, r_m, \qquad s_m = 1 - e^{g_m}, \qquad \boldsymbol{H}^k_m \leftarrow e^{g_m} \boldsymbol{H}^k_m + s_m\, \boldsymbol{k}, \quad \boldsymbol{H}^v_m \leftarrow e^{g_m} \boldsymbol{H}^v_m + s_m\, \tilde{\boldsymbol{v}}, \tag{3}$$

A strong match ($r_m \approx 1$) updates the memory decisively, while the unselected slots have $r_m = 0$, hence $e^{g_m} = 1$ and $s_m = 0$: no decay, no write, i.e. their state passes through the step *identically*. A dense recurrent state decays every entry at every step, whereas slot memories tie forgetting to writes rather than elapsed time (Zhang et al., 2024; Afzal et al., 2026); addressing those writes by pose alleviates content decay, and content fades only when observations land at the *same* address.

Reading and decoding. Reading from the memory essentially asks "what is visible from here?". The target camera is passed through the same address encoder used for writing, so the question is posed in the language the answers were filed under: the query $\boldsymbol{q}$ is scored against all slot keys and the memory returns a blend of slot contents, $\boldsymbol{o} = \mathrm{softmax}(\beta\, \boldsymbol{q} \cdot \boldsymbol{H}^k) \cdot \boldsymbol{H}^v$, with a learned per-head temperature β. The retrieved content is then rendered from the queried viewpoint: because values were canonicalized when written, the target camera re-poses the read into its own frame, effectively transporting content from the writing viewpoints to the query view. A cross-attention block lets each region of the target view pull what it needs from the readout, and a detokenizer expands the result to a coarse estimate and uncertainty (Wang et al., 2024; Wang* et al., 2025) (details in Section A.3).

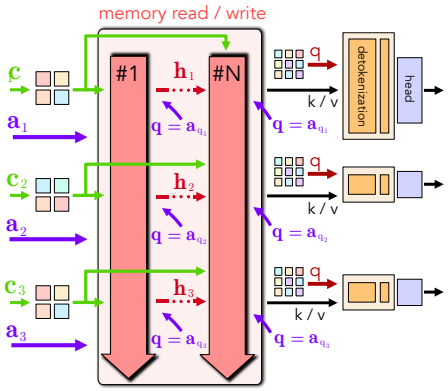


Figure 5: **SMART memory module.** Write and read paths of the slot memory.

3.3 MEMORY-CONDITIONED VIDEO GENERATION

The generator's task is to turn the memory's coarse draft into video. It is a diffusion transformer that sees the memory through an interface composed of a coarse latent and its confidence mask. The current latent, memory prediction, and its confidence mask are channelwise concatenated before patch embedding similar to Wan et al. (2025), giving each token a spatially aligned memory estimate.

Chunk-causal attention. The model generates N latent frames (a *chunk*) at a time. A fixed mask governs all blocks: a GEN token may attend to the W clean context frames before its chunk, and its own chunk bidirectionally. Each denoising step is then a compact forward over the few frames a chunk can legally see rather than the full clip, so the cost of a step depends on $(W + N)$, never on how long the video has run. For a chunk starting at frame s, the visible sequence is

$$\underbrace{x_{s-W} \cdots x_{s-1}}_{\text{CONTEXT}-\text{denoised}} \;\Big|\; \underbrace{\left(x^\tau_s,\, \hat{x}_s,\, 1{-}\tau_{c,s}\right) \cdots \left(x^\tau_{s+N-1},\, \hat{x}_{s+N-1},\, 1{-}\tau_{c,s+N-1}\right)}_{\text{GEN}-\text{noised frame, draft, and trust as channels}}$$

Further DiT, training and dataset-specific details are given in Section 4.1 and Section A.3.

3.4 TRAINING

We train in two stages, with a frozen domain-specific autoencoder throughout (Table 4). The memory is trained first, alone and as a plain regressor, so that its reads are verified retrieval before any generator depends on them. The diffusion transformer is then trained on top of the frozen memory.

Memory training. The memory is trained through a single interface: having written the observed frames $\mathcal{O}_t$ into its state, it is handed one query pose and must return the latents observed there, $\hat{z} = f_\theta(z_{\mathcal{O}_t}, \pi_{\mathcal{O}_t}; \pi_{\tau(t)}) \approx z_{\tau(t)}$. Everything the memory must learn is expressed by the choice of τ and $\mathcal{O}$: revisits (C_{rev}), answerable only by recall, co-observed novel views (C_{nvs}), answerable only by fusing stored viewpoints, past (C_{past}) and current frames (C_{cur}), and held-out spans (C_{ho}), a quarter of each clip excluded from the write and rendered purely from surrounding observations. Recall and synthesis are the same read, differing only in where the target lies relative to what was written. Reading never modifies the state, so one written prefix can be queried at arbitrarily many cameras and all targets are supervised in parallel. The recurrence itself trains as a parallel scan, without unrolling.

Objective. The loss separates what the memory must produce from how it must be addressed,

$$\mathcal{L} \;=\; \mathcal{L}^{\text{conf}}_{\text{read}} \;+\; \lambda_{\text{wf}}\, \mathcal{L}^{\text{conf}}_{\text{wf}} \;+\; \mathcal{L}_{\text{addr}}, \;\; \mathcal{L}^{\text{conf}} = \mathbb{E}\left[\boldsymbol{c}_{\tau(t)} \left(\left(\boldsymbol{x}_{\tau(t)} - \hat{\boldsymbol{x}}_{\tau(t)} \right)^2 + \epsilon \right) - \alpha \log \boldsymbol{c}_{\tau(t)} \right] \quad (4)$$

where $\hat{\boldsymbol{x}}_{\tau(t)}$ and $\boldsymbol{c}_{\tau(t)}$ are the latent and the confidence the memory predicts for the queried pose, $\boldsymbol{x}_{\tau(t)}$ the latent actually observed there. The first two terms are the confidence-aware regression loss (Wan et al., 2018), which lets the model lower its confidence where it cannot know the answer: $\mathcal{L}^{\text{conf}}_{\text{read}}$ scores the causal read, and $\mathcal{L}^{\text{conf}}_{\text{wf}}$ is a write-free pass over the finished clip's frozen state, exactly the read-only access the generator will later perform and never part of the causal rollout. The address term, $\mathcal{L}_{\text{addr}}$, keeps the code geometric without collapse: an anchor makes each token's viewing geometry recoverable from its address, and a deliberately weak VICReg spread (Bardes et al., 2022) raises the rank of the address space, while keeping the neighbourhood structure. Term definitions and weights are fleshed out in Section A.4.

Generator training. The transformer is trained with the flow-matching loss (Lipman et al., 2022) on the generation rows only. Training prepares it for two realities of rollout. The history it conditions on will be self-generated and imperfect, so the clean context is sometimes noised, with the true level stamped through the modulation. Additionally, the draft it is handed will sometimes also be wrong, so drafts are deliberately corrupted, from single cells to the whole stream. Training details and optimizer settings for both stages are listed in App. A.3.

4 EXPERIMENTS

Setup. We evaluate on *Memory Maze* (Pasukonis et al., 2022; Tunyasuvunakool et al., 2020), procedurally generated egocentric 3D mazes at 64×64, and *Minecraft* (Yan et al., 2023; Guss et al., 2019) open-terrain trajectories at 256×256 resolution (Figure 8). Memory Maze is our primary testbed where revisits are enforced by construction, the maze layout gives an unambiguous ground truth for what should be recalled, and the low resolution makes ablations affordable. Minecraft tests whether the same mechanism holds at scale, on higher resolution action-conditioned generation. For *Minecraft*, we adopt 600-frame rollouts from the training dataset of Chandratreya et al. (2026) and we generate 401-frame rollouts to train on *Memory Maze* while we evaluate at longer horizons. Memory Maze retraces the observed trajectory in reverse, whereas Minecraft continues along the action-conditioned path. Following the protocol of Chandratreya et al. (2026), these paths are sampled to close loops over the context, so reconstruction metrics primarily measure recall of observed content. The two instantiations share the interface of Section 3.3 but differ in backbone, autoencoder, and resolution (Table 4; details in Section A.5).

Metrics. For reconstruction we report mean per-frame PSNR, SSIM (Wang, 2004) with 11×11 window of $\sigma = 1.5$ and LPIPS (Zhang et al., 2018) with AlexNet features. For generation, we report FID (Heusel et al., 2017) (InceptionV3, 2048-D) and FVD (Unterthiner et al., 2018) (I3D, 1024-D).

4.1 IMPLEMENTATION

Memory design. We instantiate SMART with a GSA slot recurrence (Zhang et al., 2024) using Mamba-2 decay (Dao & Gu, 2024) per slot, adapted from Raven (Afzal et al., 2026) for pose-

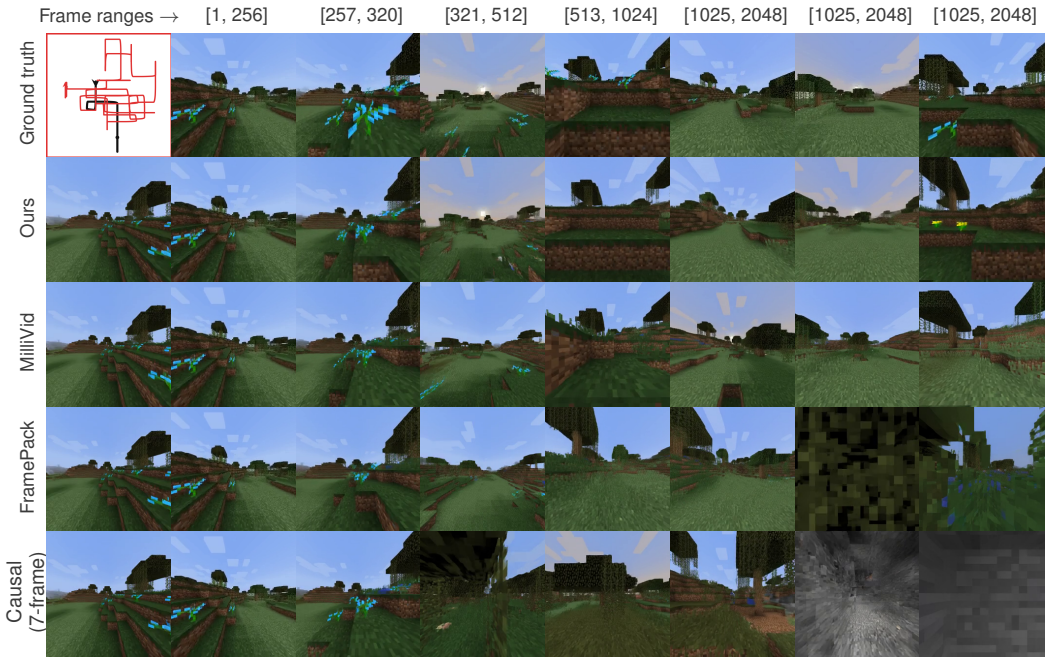


Figure 6: **Qualitative comparison on *Minecraft*.** Generated frames for each method in Table 2, on the same trajectory, at the 2048 frame horizon with 256 initial context frames.

addressed video memory. Full architectural and training details are given in Sections A.3 and A.4.

Generator design. We instantiate SMART with two diffusion-transformer backbones to evaluate the memory across different generation settings: a camera-conditioned Wan2.1-style model (Wan et al., 2025; Li et al., 2025) on *Memory Maze* and an action-conditioned DiT-B (Peebles & Xie, 2023; Chandratreya et al., 2026) on *Minecraft*. Both start from a chunk-causal generator without memory, trained by us over the full clean history on *Memory Maze* and taken from MilliVid's released 7-frame rollout checkpoint on *Minecraft*, and are then finetuned with SMART-SM. Given an observed prefix, the generator generates chunk by chunk along the control signal, and writes its own generations back into memory. Full architectural details are provided in Section A.5.

4.2 3D CONSISTENT VIDEO GENERATION

Baselines. On *Memory Maze* we compare against causal attention over the full history, its 100-frame windowed variant, DFoT (Song et al., 2025), and GeoCache, our reimplementation of the geometry-cached memory of Lyra 2.0 (Shen et al., 2026) after training faithful implementations based on their official codebases. We additionally compare against our StateSpaceDiffuser (Savov et al., 2026) reimplementation, a capacity-matched action-conditioned recurrent memory, which degrades rapidly on longer trajectories (see Section Section A.6. On *Minecraft* we instead compare against released checkpoints throughout, namely MilliVid (Chandratreya et al., 2026), the FramePack (Zhang et al., 2025) reimplementation distributed with it, and its 7-frame full-resolution rollout, each run through its authors' own sampling code. Implementation details for all baselines are given in Section A.5.

Generation results. Across both environments, SMART maintains substantially stronger long-horizon reconstruction fidelity than temporal-context baselines while remaining markedly more compute-efficient than the strongest long-context alternatives (Tables 1 and 2). On *Memory Maze*, reconstruction quality improves up to the training horizon and degrades only gradually as the trajectory grows longer, whereas all temporal-context baselines deteriorate more. Across the longer horizons, SMART maintains a clear margin in reconstruction quality while requiring substantially less compute than DFoT and GeoCache. On *Minecraft*, SMART achieves the highest PSNR and SSIM across all horizons and the best LPIPS in three of four bands, while trailing MilliVid on FID and on FVD at intermediate horizons, at roughly $2.7\times$ lower compute. Qualitative examples are shown in Figure 13 for *Minecraft* and Figure 10 for *Memory Maze*.

Table 1: **3D consistent video generation on *Memory Maze*.** Methods generate the reverse leg from the observed forward trajectory in parantheses. Column groups are total trajectory length. FID pools a band's frames, FVD its 16-frame clips.

METHOD	$(201+21)+201=423$ *frames*						$(401+21)+401=823$ *frames*					
	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓
CAUSAL (100F)	**0.096**	25.0	0.809	0.131	6.39	4.06	**0.193**	21.8	0.739	0.198	8.85	7.19
CAUSAL (FULL)	0.117	30.0	0.891	0.034	5.36	2.13	0.299	24.3	0.789	0.100	7.96	4.61
GEOCACHE (GT-D)	0.454	24.6	0.811	0.098	6.25	3.57	0.936	24.6	0.810	0.095	6.03	2.52
DFOT	0.566	**32.0**	**0.919**	**0.030**	5.74	**1.89**	1.130	28.4	0.865	0.079	5.54	1.89
SMART	0.108	31.1	**0.919**	0.040	**5.22**	2.25	0.213	**29.9**	**0.900**	**0.046**	**5.08**	**1.83**

METHOD	$(801+21)+801=1623$ *frames*						$(1601+21)+1601=3223$ *frames*					
	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓
CAUSAL (100F)	**0.394**	20.4	0.709	0.227	10.3	8.88	**0.824**	19.1	0.677	0.266	11.6	11.1
CAUSAL (FULL)	0.850	20.8	0.718	0.210	47.4	32.6	2.727	18.3	0.669	0.351	158.5	199.6
GEOCACHE (GT-D)	1.872	25.3	0.822	0.080	5.71	2.04	3.772	25.7	0.827	0.073	5.67	1.76
DFOT	2.259	26.6	0.838	0.103	5.48	**1.60**	4.518	24.0	0.795	0.147	5.72	1.80
SMART	0.430	**29.3**	**0.890**	**0.051**	**5.03**	1.66	0.900	**27.6**	**0.863**	**0.070**	**5.00**	**1.76**

Table 2: **3D consistent video generation on *Minecraft*.** Methods generate frames 257–2048 from 256 ground-truth context frames. Column groups are distance past the context. FID pools a band's frames, FVD its 16-frame clips.

METHOD	*1–64 frames*						*65–256 frames*					
	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓
CAUSAL (7F)	**0.72**	17.18	0.583	0.325	**43.36**	64.37	**2.62**	11.67	0.437	0.551	47.78	182.03
FRAMEPACK	1.71	19.13	0.612	0.260	44.66	56.67	6.62	14.04	0.498	0.448	48.67	112.84
MILLIVID	2.40	21.16	0.615	0.196	44.68	47.83	8.95	18.43	0.562	**0.284**	**43.34**	**44.28**
SMART	0.91	**21.96**	**0.647**	**0.187**	44.60	**45.86**	3.46	**18.54**	**0.571**	0.294	44.01	56.00

METHOD	*257–768 frames*						*769–1792 frames*					
	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓	PFLOPs↓	PSNR↑	SSIM↑	LPIPS↓	FID↓	FVD↓
CAUSAL (7F)	**7.77**	10.99	0.412	0.627	70.19	507.07	**18.1**	10.51	0.392	0.700	109.60	963.68
FRAMEPACK	19.7	11.97	0.444	0.540	58.12	228.78	45.9	11.21	0.411	0.601	64.31	333.38
MILLIVID	26.2	16.05	0.522	0.371	**41.19**	**56.37**	65.5	13.84	0.468	0.477	**38.85**	104.89
SMART	10.2	**16.38**	**0.537**	**0.370**	44.55	72.61	23.8	**14.85**	**0.503**	**0.446**	43.33	**101.75**

5 ANALYSIS & ABLATIONS

Slot memory addressing by region tokens. Our slot memory addressing mechanism via region tokens allows for the decoupling of elapsed time and forgetting by writing only during revisits. In Figure 7, we show that our design creates a strong overlap between the camera pose of a frame (a) and the memory slot to which it is being written (b). As a result, a slot only decays when re-routed. Reading the memory is a softmax-attention over the mixture of camera pose encodings, yielding a convex combination of write tokens, meaning that each query is attributable to the source frames. Accordingly, Figure 7 shows that slots holding the query's place draw $1.7\times$ uniform mass (c). Finally, we observe that not all heads behave similarly: In Figure 7, we observe that while some heads are localizing heads for retrieving local information (e), others preserve a global attention pattern (f).

Ablations. Table 3 varies one component of SMART's memory module at a time, decoding frames from memory alone at queried poses. Due to perception-distortion trade-off (Blau & Michaeli, 2018) and lack of generative stochasticity, the memory module alone can achieve high reconstruction scores at worse perceptual quality. Replacing the routed slots with a dense state of the same size matches SMART at training length but degrades sharply beyond it, whereas SMART holds its quality out to eight times that horizon: sparse writes allow extending memory horizon as a slot without further writes is left intact. Removing the SE(3) value transform costs quality at every horizon, indicating that stored content must be expressed relative to the querying camera to be read back from a new viewpoint. As expected, halving the memory heads reduces the retrieval capability in longer sequences.

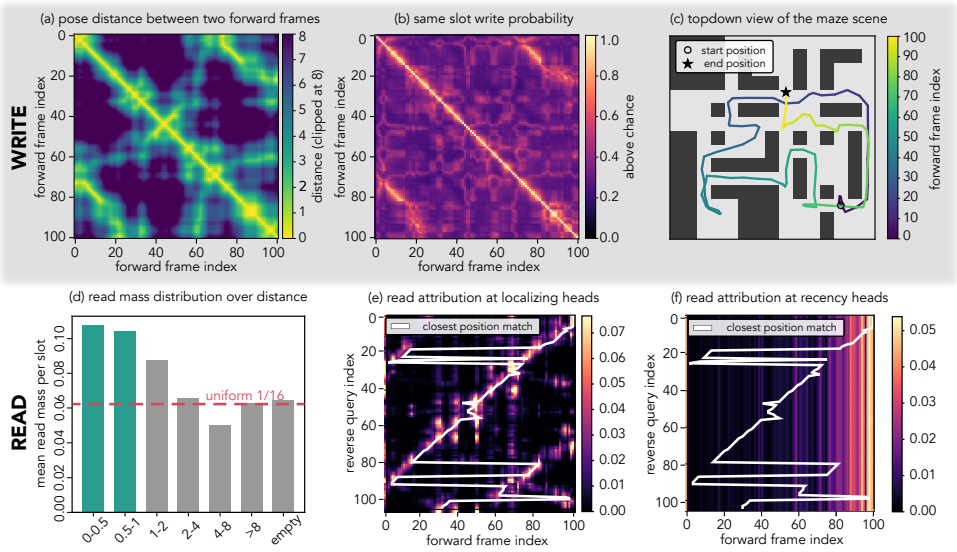


Figure 7: **What the slot memory stores, and how a read recovers it.** SMART-SM on 401-frame maze episodes: the forward leg is written, the reverse leg read from the state. **Write.** (a) Euler distance between two forward frame positions and (b) the probability that their region tokens land in the same slot. **Read.** (d) Mean read mass per slot against the distance from the query to the content that slot currently holds where content position is the decayed average of all content written to respective slot. (e, f) Attribution of each reverse read to the forward frames in memory slots where the white trace is the pose-nearest forward frame.

Table 3: **Memory Only ablation results.** Reconstruction quality on *Memory Maze*, changing one component of SMART at a time. SMART applies the SE(3) value transform to the 64 value channels of the head. All models are trained on 401 frame episodes.

VARIANT	*401 frames*			*1601 frames*			*3201 frames*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SMART-SLOT MEMORY	32.7	**0.940**	0.024	**32.4**	**0.936**	**0.024**	**31.9**	**0.931**	**0.026**
NO SE(3) VALUE WARPING	31.5	0.924	0.031	30.0	0.906	0.041	29.6	0.900	0.045
SMART-DENSE MEMORY	**32.9**	0.939	**0.020**	26.5	0.852	0.074	21.5	0.756	0.191
SCALING DOWN HEADS (/2)	32.2	0.935	0.028	24.5	0.810	0.146	22.9	0.772	0.192

6 CONCLUSIONS

In this work we presented SMART, a pose-addressed slot memory for long-horizon video generation that combines constant-size recurrent storage with spatially indexed recall. By separating pose-derived addressing from content storage, updating only the slots of the observed region, and canonicalizing stored values across viewpoints, SMART makes retention depend on where content was seen rather than when. The retrieved view serves as a coarse draft that a chunk-causal diffusion model refines into video, drawing on stored scene content where it is available and on its generative prior elsewhere. Across *Memory Maze* and *Minecraft*, this preserves scene-specific consistency over substantially longer horizons than temporal-context baselines while avoiding the growing storage and reconstruction dependence of explicit geometric memories.

Limitations & Future Work. A central limitation of the current design is that fixed memory capacity bounds the spatial extent a model can retain. Although selective routing prevents information from decaying with elapsed time, sufficiently large environments eventually force different regions to reuse the same slots, causing previously stored content to be overwritten and forgotten. A natural direction for future work is therefore to make the memory capacity adaptive at inference time, for example, by allocating additional slots or memory banks online when the existing state becomes saturated, while preserving efficient pose-based addressing and retrieval. In addition, our current experiments use relatively modest generator backbones, scaling the approach to substantially larger video models is an important next step to assess how the memory mechanism interacts with stronger generative priors and higher visual fidelity.

AI USE STATEMENT

We used generative AI tools to assist with implementing and debugging research code, provide feedback on experimental methodology, identify relevant literature and revise the manuscript. The authors reviewed and revised the AI-assisted text, checked references against the original sources and reviewed and tested AI-assisted code. We take responsibility for the final manuscript, methods, results, and accompanying code, including all content produced with generative AI assistance.

REFERENCES

Arshia Afzal, Aviv Bick, Eric P. Xing, Volkan Cevher, and Albert Gu. Raven: High-recall sequence modeling with sparse memory routing. arXiv preprint, 2026.

Mohammad Asim, Christopher Wewer, and Jan Eric Lenssen. Scenetok: A compressed, diffusable token space for 3d scenes. *arXiv preprint arXiv:2602.18882*, 2026.

Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David B. Lindell, Zan Gojcic, Sanja Fidler, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra: Generative 3d scene reconstruction via video diffusion model self-distillation. In *International Conference on Learning Representations (ICLR)*, 2026.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6228–6237. IEEE, 2018.

Ishaan Preetam Chandratreya, David Charatan, Basile Van Hoorick, Sergey Zakharov, Vitor Guizilini, Phillip Isola, and Vincent Sitzmann. Millivid: Hierarchical latents for long-range consistency in video generation. In *arXiv*, 2026. URL https://davidcharatan.com/millivid.

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.

Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. In *International Conference on Learning Representations*, volume 2026, pp. 50694–50718, 2026a.

Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, Yuyang Zhao, Huizi Mao, Ying-Cong Chen, Enze Xie, Xiaojuan Qi, and Song Han. Longlive2.0: An nvfp4 parallel infrastructure for long video generation. *arXiv preprint arXiv: 2605.18739*, 2026b.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Zhang Tao, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 106613–106631, 2026.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

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

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Chunyang Li, Yuanbo Yang, Jiahao Shao, Hongyu Zhou, Katja Schwarz, and Yiyi Liao. Rerope: Repurposing rope for relative camera control. SIGGRAPH Conference Papers '26, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725548.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *Advances in Neural Information Processing Systems*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. Gta: A geometry-aware attention mechanism for multi-view transformers. In *International Conference on Learning Representations (ICLR)*, 2024.

Jurgis Pasukonis, Timothy Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. *arXiv preprint arXiv:2210.13383*, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8733–8744, October 2025.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6121–6132. IEEE, 2025.

Nedko Savov, Naser Kazemi, Deheng Zhang, Danda Pani Paudel, Xi Wang, and Luc V Gool. Statespacediffuser: Bringing long context to diffusion world models. *Advances in Neural Information Processing Systems*, 38:68865–68898, 2026.

Tianchang Shen, Sherwin Bahmani, Kai He, Sangeetha Grama Srinivasan, Tianshi Cao, Jiawei Ren, Ruilong Li, Zian Wang, Nicholas Sharp, Zan Gojcic, et al. Lyra 2.0: Explorable generative 3d worlds. *arXiv preprint arXiv:2604.13036*, 2026.

Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. In *Forty-second International Conference on Machine Learning*, 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric and challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Sheng Wan, Tung-Yu Wu, Wing H Wong, and Chen-Yi Lee. Confnet: predict with confidence. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2921–2925. IEEE, 2018.

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

Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. In *2025 International Conference on 3D Vision (3DV)*, pp. 78–89. IEEE, 2025.

Qianqian Wang*, Yifei Zhang*, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *CVPR*, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

Weijie Wang, Haoyu Zhao, Yifan Yang, Feng Chen, Zeyu Zhang, Yefei He, Zicheng Duan, Donny Y. Chen, Yuqing Yang, and Bohan Zhuang. Latent spatial memory for video world models. *arXiv preprint arXiv:2606.09828*, 2026.

Zhou Wang. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Zhengxuan Wei, Xu Guo, Xinghui Li, Xunzhi Xiang, Min Wei, Yiran Zhu, Qiulin Wang, Xintao Wang, Pengfei Wan, Xiangwang Hou, et al. Geometry-aware implicit memory for video world models. *arXiv preprint arXiv:2606.02436*, 2026.

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory. *Advances in Neural Information Processing Systems*, 38:49632–49652, 2026.

Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent transformers for video generation. In *International Conference on Machine Learning*, pp. 39062–39098. PMLR, 2023.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Wei Yu, Runjia Qian, Yumeng Li, Liquan Wang, Songheng Yin, Sri Siddarth Chakaravarthy P, Dennis Anthony, Yang Ye, Yidi Li, Weiwei Wan, and Animesh Garg. Mosaicmem: Hybrid spatial memory for controllable video world models. *arXiv preprint arXiv:2603.17117*, 2026.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

Jinjing Zhao, Fangyun Wei, Zhening Liu, Hongyang Zhang, Chang Xu, and Yan Lu. Spatia: Video generation with updatable spatial memory. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 2026.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming visual geometry transformer. In *International Conference on Learning Representations*, volume 2026, pp. 88055–88072, 2026.

A Appendix

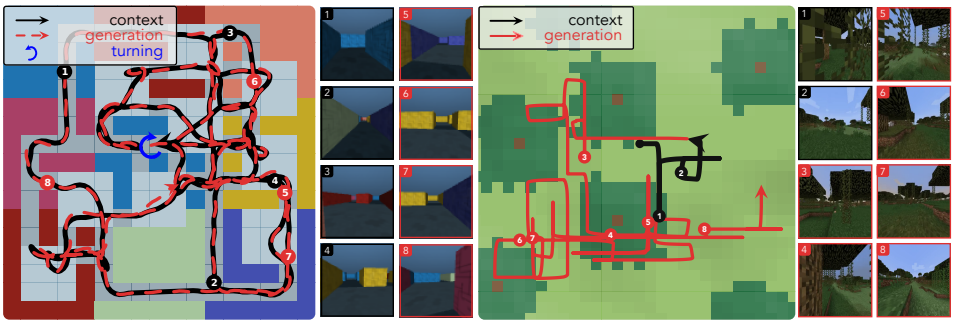
A.1 Relation to Linear Recurrent Models

Closely related to our work, we draw similarities and differences with Raven (Afzal et al., 2026), a gated-slot attention based linear-time sequence model. We build on its state architecture where each head keeps a fixed number of slots, updated by the gated slot attention recurrence (Zhang et al., 2024) with a learned decay, and a router assigns every token to the slots it writes, so the state stays constant in size however long the sequence grows. Our implementation leverages the same underlying codebase from `fla` Yang & Zhang (2024) (Section A.3). The difference lies in what the memory is asked to do. In recurrent sequence models more broadly, queries, keys, values and routing are all projections of the current token's hidden state, and the state is read once, to predict the next token.

A camera-conditioned generator instead has to read the scene from any viewpoint, past or future, independently of the last frame written. We therefore derive keys, queries and routing from the camera pose alone and values from the frame content to facilitate learning of pose-aligned memory slots rather than content similarity-based associations. Each token writes a single slot, which remains stable until that place is observed again, and values are stored in a camera-independent frame to be re-posed by the target camera on read (Section A.3). Finally, the output is not a next-token prediction but a draft of a target view, read from a state while the generator denoises and updates the state once the chunk is complete.

A.2 Datasets

Memory Maze. Measuring retrieval needs trajectories that return to what they have already seen, which the released *Memory Maze* episodes (Pasukonis et al., 2022) cannot guarantee: they record an RL agent exploring freely. We therefore roll out our own evaluation episodes in the same 15×15 environment on MuJoCo (Todorov et al., 2012) and `dm_control` (Tunyasuvunakool et al., 2020), with an imposed three-leg structure of a forward leg of N frames, a 21-frame in-place turn, and a return leg of N frames, so an episode is $2N+21$ frames long (Figure 8a). The forward leg is a stochastic breadth-first coverage sweep, repeatedly heading for the nearest cell the route has not yet entered rather than following a shortest path, and the return leg then retraces the positions actually walked. Frames are 64×64 RGB from a rigidly mounted first-person camera.



a) Memory Maze dataset sample topdown view b) Minecraft dataset sample topdown view

Figure 8: **Evaluation trajectories.** Top-down view of one episode per dataset: (a) a *Memory Maze* forward–turn–return rollout, and (b) a *Minecraft* continuation rollout.

Each frame stores the agent position and heading. Camera extrinsics and intrinsics are derived from the pose at load time. For training we re-render the released Memory Maze episodes (Pasukonis et al., 2022) through the same camera (29,000 train and 1,000 evaluation), and cut into overlapping 401-frame snippets so that the environment's goal spheres and target rings are hidden.

Minecraft. MilliVid's *LoopCraft* (Chandratreya et al., 2026) provides our training data and the evaluation protocol we adopt, but its released test split is 1,024 frames long and our claim concerns horizons beyond that. We therefore generate a 2,048-frame split with their released generator to collect MineRL (Guss et al., 2019) trajectories through the pipeline of TECO (Yan et al., 2023) where the agent moves forward or turns left/right, rendered at 512×512 and downsampled to 256×256. Unlike our Memory Maze episodes these rollouts are continuations with no turn and no return leg, the model being given 256 context frames and generating the remaining 1,792 (Figure 8b). We keep Chandratreya et al. (2026)'s loop-closure bias, drawing 300 candidate action sequences per episode, scoring each by the Chamfer distance between the planned positions of the first 256 frames and the remainder, and replaying one sampled uniformly from the ten scoring lowest.

MilliVid's *LoopCraft* stores each episode's video, a one-hot action per frame, and the agent position and heading read directly from the simulator. To achieve pose-conditioning, we derive the camera poses based on the specifications of Guss et al. (2019) and agent's pose. For training we use 601-frame rollouts, which is sufficient for the long-horizon extrapolation we evaluate.

A.3 SMART Implementation Details

Tokenization. Each latent frame is cut into $P = H_z \cdot W_z / p^2$ patches of size p and embedded by a strided convolution,

$$\boldsymbol{e}_{t,i} = \mathrm{Emb}(\boldsymbol{z}_t)_i \in \mathbb{R}^D, \qquad i = 1, \ldots, P, \tag{5}$$

which at $p{=}1$ is one token per latent cell. A tokenization module then compresses the frame's patch grid into $J \ll P$ *write tokens*: J learned queries attend the P patch tokens, whose keys carry a two-dimensional rotary embedding over the (H_z, W_z) grid (Heo et al., 2024), and a causal self-attention over the J queries makes every prefix $\boldsymbol{c}_{t,1:j}$ a self-contained code of the frame (Asim et al., 2026),

$$\mathcal{C}_t = [\boldsymbol{c}_{t,1}, \ldots, \boldsymbol{c}_{t,J}] = \phi_{\text{content}}(\boldsymbol{e}_{t,1}, \ldots, \boldsymbol{e}_{t,P}), \qquad \boldsymbol{c}_{t,j} \in \mathbb{R}^D. \tag{6}$$

Write tokens are laid out frame-major into one sequence of length $S = TJ$, and the memory values are a linear projection of them, split across the N_h heads,

$$\boldsymbol{v}_{h,t,j} = (\boldsymbol{W}_v \, \boldsymbol{c}_{t,j})_h \in \mathbb{R}^d. \tag{7}$$

$\boldsymbol{c}_{t,j}$ carries neither a 3D-aware nor a temporal tag of its own via positional encodings so that the temporal order is represented only by the recurrence itself.

Memory implementation. Our recurrent memory implementation uses the chunked-update GSA kernels of the `fla` package (Yang & Zhang, 2024), inspired from Raven (Afzal et al., 2026). In

contrast to Raven, which derives keys, queries, router logits and values from the same layer input for language modeling, our design separates the pose-derived addressing pathway (keys, queries, router) from the content-derived value pathway, and writes 16 tokens per frame rather than one per position based on the tokenization approach we took in Section 3.2. Routing is top-1 with the raw sigmoid score of the winning slot as write amplitude, so the write gate is not renormalized across slots. Read-only operations are implemented by running the recurrence with zero keys, values and gates, which leaves the state unchanged.

Value transport. A view depends on where it was seen from, but a memory of the scene should not: values are therefore written in a shared camera-independent frame, the writing camera's inverse acting on their channels as an SE(3) transform, and re-posed by the target camera at read time (Miyato et al., 2024; Li et al., 2025). Because the slot read is linear in the values, a value written at π_τ and read at π^* undergoes exactly the relative transform $\rho(\pi^*)\rho(\pi_\tau)^{-1}$, as if carried directly from the old camera to the new one. Keys are deliberately not transformed. A slot key is an exponential moving average over many writes, so no single writing camera exists to unrotate, and the relative-camera effect the rotation would provide already reaches the read through the values, so the matching stays absolute by design.

VAE design. Both autoencoders are frozen before any memory or generator is trained, and within a dataset a single autoencoder serves every latent-space method, so that all reported pixel metrics are measured against a common ceiling.

On *Memory Maze* we finetune convolutional Wan-VAE (Wan et al., 2025) and leave its architecture untouched with 16 latent output channels and causal 3D convolutions giving $8\times$ spatial and $4\times$ temporal compression. Encoder and decoder are both trained for $40{,}000$ steps on the re-rendered episodes of Section A.2 according to the training details of Wan et al. (2025).

On *Minecraft* we use MilliVid's released autoencoder (Chandratreya et al., 2026) unmodified, encoder and decoder alike. It is a per-frame vision transformer rather than a convolutional video model over 256×256 inputs at patch size 16, emitting 32 channels on a 16×16 grid, with a four-level pyramid of 256, 64, 16 and 4 tokens per frame. It applies *no* temporal compression, so a rollout keeps as many latent frames as it has observations and the memory is written once per frame. Our memory and generator consume only the finest level of that pyramid with 256 tokens per frame.

DiT design. To show that the memory is not tied to one generator, we instantiate it with two diffusion transformers: a camera-conditioned Wan2.1-style model (Wan et al., 2025) with 250M parameters on *Memory Maze* following the model size of Po et al. (2025) and MilliVid's action-conditioned DiT-B (Peebles & Xie, 2023; Chandratreya et al., 2026) with 130M parameters on *Minecraft*. Both read the memory through channelwise concatenation onto its input channels and every rollout step generates chunks of $N{=}3$ latent frames from a window of $W{=}6$ clean frames. Everything else follows the respective backbone (Table 4).

Memory Maze. The backbone is a stack of 14 Wan2.1 (Wan et al., 2025) blocks of width $1{,}024$ with 8 heads, conditioned only through Wan's per-token timestep modulation, without cross-attention. A 36-channel patch embedding maps each cell of the 8×8 latent grid to one token, 64 per frame. The camera enters attention as PRoPE (Li et al., 2025), a parameter-free camera projection on queries and keys on 40 head channels, while the remaining carry temporal and 2D-rotary embeddings. The sequence is laid out as $[\,\textsc{sink} \mid \textsc{context} \mid \textsc{gen}\,]$: frame 0 is a permanent clean sink token group, the context rows hold the W clean frames before the chunk, and the generation rows hold the noised chunk, which attends to the anchor, its window and itself. Thus, a generation token then attends to at most $(1 + W + N) \times 64 = 640$ tokens.

Minecraft. The backbone is MilliVid's DiT-B (Chandratreya et al., 2026), 12 blocks of width 768 with 12 heads, reproduced parameter for parameter. Conditioning is per token: every token carries its own adaptive-layer-norm modulation from its noise level, the action embedding (forward, turn left/right) is added to that modulation, and position is an additive sinusoidal code over frame. Each of the 256 cells of the 16×16 latent grid is one token, embedded by a linear map over 65 input channels; the 33 columns beyond the latent are the memory interface and the only parameters absent from the released checkpoint. Attention is over the $W + N = 9$ frames of a clip, $2{,}304$ tokens against the $3{,}840$ of MilliVid or FramePack, and there is neither camera input nor classifier-free guidance.

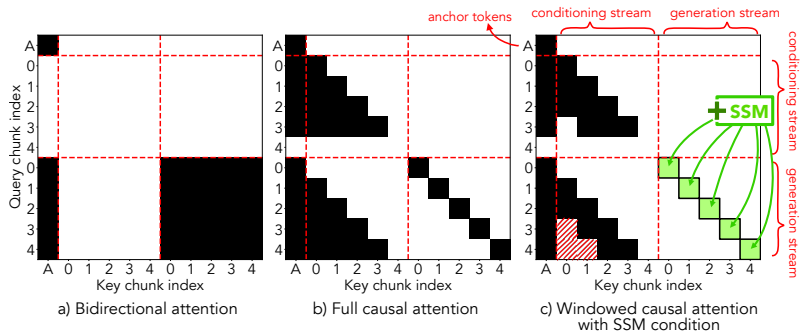


Figure 9: **Attention masks.** Allowed attention between query and key chunks for the generators in this work: (a) the bidirectional pretraining of Section A.4, (b) the full-history causal DiT of the Causal (Full) arm, (c) the windowed chunk-causal attention of SMART, in which a generation chunk reads the W clean frames before it and the memory's draft. Anchor tokens are the clean frame 0.

Table 4: **Instantiation of SMART on the two datasets.** The memory interface, chunk size and context window are shared; everything else follows the backbone each dataset inherits.

	Memory Maze	*Minecraft*
Backbone	Wan2.1-style DiT (Wan et al., 2025)	DiT-B (Peebles & Xie, 2023; Chandratreya et al., 2026)
Parameters	250M	130M
Blocks × width × heads	$14 \times 1{,}024 \times 8$	$12 \times 768 \times 12$
Control signal	camera pose (PRoPE)	discrete action for modulation
Positional code	temporal and 2D rotary, camera PRoPE	sinusoidal over frame, row and column
Autoencoder	Wan2.1 VAE, finetuned	MilliVid hierarchical autoencoder, pretrained
Frame resolution	64×64	256×256
Latent frame	$16 \times 8 \times 8$, $4\times$ temporal compression	$32 \times 16 \times 16$, per frame
Tokens per frame	64	256
Training clip	401 frames	601 frames
Chunk N / window W	3 / 6	3 / 6
Tokens per denoising step	640	2,304
Objective	rectified flow, v-prediction	shifted-cosine VP, v-prediction
Noise-level draw	logit-normal	uniform
Sampler	35 Euler steps	50 DDIM steps
Optimizer	AdamW	AdamW
Initialization	causal DiT	causal DiT from Chandratreya et al. (2026)
Memory module	64.6M weights, 98,304-float state	132.8M weights, 98,304-float state

Memory conditioning. The generator reads the memory as a draft rather than through attention. Before a chunk is denoised, the frozen memory is queried at the pose of each target frame and returns a draft latent with a per-cell confidence $\boldsymbol{c} \geq 1$, which becomes the trust value $1 - \tau_c$ with

$$\tau_c = \mathrm{clip}\left(\frac{\alpha/\boldsymbol{c} - \epsilon}{\alpha - \epsilon},\ 0,\ \tau_{\max}\right). \tag{8}$$

The numerator is the expected squared error at the stationary point of Equation (4), the denominator its largest value at $\boldsymbol{c} = 1$, and $\tau_{\max} = 0.8$. As in Wan2.1 image-to-video (Wan et al., 2025), the draft is concatenated to the noisy latent along channels as [latent ; trust ; draft], so that each draft cell is aligned with the cell it predicts and no tokens are added. Context frames place their own clean latent in the draft slot at trust 1. On *Memory Maze* this gives $16 + 4 + 16 = 36$ input channels, the trust being repeated over the 4 frames folded into each latent frame, and on *Minecraft* $32 + 1 + 32 = 65$, since its autoencoder has no temporal compression. Once denoised, the chunk is written into the memory for autoregressive generation.

A.4 TRAINING FORMULATION

Spatially-Structured Memory Training. The memory is trained alone and before any generator, as a deterministic regressor over posed latents. For each training clip (401 and 601 frames), we pre-encode the frames with a pretrained VAE into 101 and 601 latent frames for Memory Maze and Minecraft datasets, respectively. The camera poses are kept at the native frame rate, rebased to $\boldsymbol{E}_0 = \boldsymbol{I}$, and divided by a single scene-scale constant. At every latent position t, the model writes the observable prefix $\mathcal{O}_t$ into its state, is handed one query pose $\pi_{\tau(t)}$, and returns the latents observed

for the query pose, $\hat{\boldsymbol{z}}_t = f_\theta(\boldsymbol{z}_{\mathcal{O}_t}, \pi_{\mathcal{O}_t};\, \pi_{\tau(t)}) \approx \boldsymbol{z}_{\tau(t)}$. Reading operation never modifies the state, so queries are executed at every position in parallel since the recurrence itself trains as a parallel scan, without unrolling (Zhang et al., 2024).

Target selection. The target index $\tau(t)$ is drawn independently at each position. A *revisit*, $\mathcal{C}_{\mathrm{rev}}$, is a future frame whose camera lies within 0.5 world units and a camera direction with cosine similarity larger than 0.9 that is already in the written prefix, and is answerable only by recalling that visit (10% of targets). A *fusion* target, $\mathcal{C}_{\mathrm{nvs}}$, is a future pose that is *not* a near-duplicate of anything written, but whose viewing footprint is covered by at least one written camera (45%). It is answerable only by fusing several stored viewpoints. The *current* frame, $\mathcal{C}_{\mathrm{cur}}$, reads its own write back out, supervising the write–read round trip end to end (2%). The remaining 42% recall a uniformly random previously written frame ($\mathcal{C}_{\mathrm{past}}$) that is re-queried.

Held-out spans. Independent of the draw above, six blocks of four consecutive latent frames, 24 of the 101, are reserved at every step and never written ($\mathcal{C}_{\mathrm{ho}}$). Reserved frames are withheld as revisit evidence and as fusion observers, and each claims two later positions as queries, so about half of all queries ask for a span the memory was never shown and must render from the observations surrounding it. These positions are scored by a plain squared error rather than the confidence-weighted.

Objective. We instantiate Equation (4) with $\alpha = 0.2$ and $\epsilon = 0.05$. The two regression terms differ only in τ and in which state is read: $\mathcal{L}^{\mathrm{conf}}_{\mathrm{read}}$ uses the sampled target above, while $\mathcal{L}^{\mathrm{conf}}_{\mathrm{wf}}$ sets $\tau(t) = t$ and re-reads every frame at its own pose from the frozen final rollout state, which makes it a scene summary.

The address term is $\mathcal{L}_{\mathrm{addr}} = \lambda_{\mathrm{pose}}\mathcal{L}_{\mathrm{pose}} + \lambda_{\mathrm{var}}\mathcal{L}_{\mathrm{var}} + \lambda_{\mathrm{cov}}\mathcal{L}_{\mathrm{cov}}$. The anchor asks a per-head *affine* map $(\boldsymbol{W}^p_h, \boldsymbol{b}^p_h)$ to recover the viewing geometry of its own tile under each of the f cameras the latent frame folds,

$$\begin{aligned} \mathcal{L}_{\mathrm{pose}} &= \mathbb{E}_{h,t,j,i}\Big[\tfrac{1}{9}\big\|\big(\boldsymbol{W}^p_h \boldsymbol{a}_{h,t,j} + \boldsymbol{b}^p_h\big)_i - \boldsymbol{y}_{t,j,i}\big\|_2^2\Big], \\ \boldsymbol{y}_{t,j,i} &= \Big[\,\bar{\boldsymbol{d}}_{i,j}\;;\; \big\{(\boldsymbol{p}_i + \lambda\,\bar{\boldsymbol{d}}_{i,j})/\sigma\big\}_{\lambda\in\{4,\,16\}}\,\Big] \in \mathbb{R}^9, \end{aligned} \tag{9}$$

where i ranges over the f cameras of the fold, $\boldsymbol{p}_i$ is the camera centre and $\bar{\boldsymbol{d}}_{i,j}$ the mean unit ray of tile j under camera i. Collecting the $N{=}BTJ$ addresses of a batch at head h into a centred $\boldsymbol{Z}_h \in \mathbb{R}^{N\times d_a}$, one row per address and d_a (the per-head address channels), the spread is the VICReg variance and covariance pair (Bardes et al., 2022),

$$\mathcal{L}_{\mathrm{var}} = \mathbb{E}_{h,p}\left[\max\Big(0,\, 1 - \sqrt{\mathrm{Var}(\boldsymbol{Z}^p_h)}\Big)\right], \quad \mathcal{L}_{\mathrm{cov}} = \mathbb{E}_h\left[\tfrac{1}{d_a}\textstyle\sum_{p\neq q}\big[C(\boldsymbol{Z}_h)\big]^2_{pq}\right], \tag{10}$$

where $\boldsymbol{Z}^p_h \in \mathbb{R}^N$ is the p-th column of $\boldsymbol{Z}_h$ (i.e., coordinate p of every address in the batch) and $C(\boldsymbol{Z}_h)$ is its $d_a \times d_a$ covariance. We use $\lambda_{\mathrm{wf}} = \lambda_{\mathrm{pose}} = 1$, $\lambda_{\mathrm{var}} = 0.1$ and $\lambda_{\mathrm{cov}} = 0.001$.

Optimization. We use AdamW (Loshchilov & Hutter, 2017) with $\beta = (0.9, 0.95)$, a learning rate of $5e{-}4$ decayed by a cosine schedule, and a weight decay of 0.01. Gradient checkpointing is applied to the recurrent blocks, the write-free reads and the camera encoder to keep training feasible on 12 RTX4090 GPUs for $100{,}000$ iterations with an effective batch size of 72.

3D Consistent Video Generation Training. The generator is trained after the memory module is trained according to Sections A.4 and 3.2 and on top of it: the memory stays frozen for storing the latent history and to condition the transformer in a causal manner. It predicts the velocity with one noise level per chunk of $N{=}3$ latent frames and the $W{=}6$ clean frames before the chunk as context, and the two augmentations of Section 3.4 are applied at the rates given below. Half of the clips have each context frame noised with probability 0.5 to a level drawn from $\mathcal{U}(0, 0.3]$, and each draft frame is corrupted with probability 0.08, while the whole draft is replaced by noise for 20% of clips on *Memory Maze* and 5% on *Minecraft*, where a further 15% of clips instead have their context replaced by noise so that the draft is the only source of information.

Memory Maze DiT training. No pretrained generator exists for *Memory Maze*, so we train one in two stages, the second initialized from the first. (i) A bidirectional image-to-video model is pretrained on 101-latent-frame clips with all-to-all attention (Figure 9a) and one noise level per clip, for $50{,}000$ steps at an effective batch of 96 on 12 RTX4090 GPUs, with a learning rate of $5e{-}4$ and decayed

by a cosine schedule. (ii) It is then finetuned with the frozen memory: the context is cut to the $W{=}6$ window of Figure 9c and the draft and trust channels are attached. Noise levels are drawn logit-normally per chunk and the target is the rectified-flow velocity.

We use AdamW (Loshchilov & Hutter, 2017) with $\beta = (0.9, 0.95)$, a weight decay of 0.01, and a learning rate of $1e{-}4$ decayed by a cosine schedule. The model is finetuned for $15{,}000$ steps at an effective batch of 40. At inference every chunk is denoised in 35 Euler steps and every generated chunk is written back into the memory module. The Causal (Full) baseline is a separate finetune of (i) with the full-history chunk-causal mask of Figure 9b, in which every chunk attends to the clean ground-truth history below it, trained for $\approx 40{,}000$ steps at batch 72 with a learning rate of $1e{-}4$ and cosine decay scheduling.

Minecraft DiT training. The generator is initialized from MilliVid's released 7-frame full-resolution rollout checkpoint, the Causal (7F) arm of Table 2. Only the patch embedding is widened, from 32 to 65 input channels for the memory interface. The target is the v-prediction of a shifted-cosine variance-preserving schedule with uniformly drawn noise levels. Each step loads a full 601-frame window, draws one boundary $s \sim \mathcal{U}[6, 598]$, prefills the memory with the ground-truth frames $[0, s)$ and reads the draft of frames $[s, s{+}3)$ from that state without writing, so the draft depends only on frames before the chunk. The trust label is the same map as for *Memory Maze* with constants, $\alpha = 0.15$ and $\epsilon = 0.007$, and $\tau_{\max} = 0.8$. We use AdamW (Loshchilov & Hutter, 2017) with a weight decay of 0.01 and a learning rate of $1.5e{-}4$. The model is trained for $60{,}000$ steps at an effective batch of 64 with 4 accumulation steps. At inference, the memory is prefilled with the 256 context frames and each 3-frame chunk is denoised with 50-step DDIM sampler and written back into the memory.

A.5 3D Consistent Video Generation Evaluation

Memory design. The SMART-SM is instantiated identically on both datasets, as described in Section 3.2 and Section A.3 with 8 layers of 6 heads, head dimension 64, $M{=}16$ slots per head, top-1 routing whose sigmoid write amplitude is left unnormalized, a write gate tied to the Mamba-2 decay, the SE(3) value warp acting on all head channels as detailed in Section A.3. Keys and queries are functions of pose alone, and each frame contributes 16 write tokens. The state is therefore $2 \times L \times H \times M \times d = 98{,}304$ floats on both datasets and at every horizon.

Baseline Implementation Details. Every baseline is generated based on its authors' implementation and scored by us after adapting for the benchmarks. On *Memory Maze*, the GeoCache (based on Lyra2.0 (Shen et al., 2026)) and DFoT (Song et al., 2025) architecture, diffusion backend, sampler, sliding-window schedule and guidance belong to the reference implementation, and evaluated on our test dataset after training models from scratch. On *Minecraft*, the rollout is produced by the released code unchanged. All methods are run on the same episodes and decoded by a shared autoencoder.

Causal DiT. On *Memory Maze*, Causal (Full) is the full-history causal finetune of our bidirectional model (Section A.4), scored as an arm of its own with the same 35-step rollout and key–value cache, and Causal (100F) is the same recipe with the clean context capped at 25 latent frames (100 pixel frames). On *Minecraft*, Causal (7F) is MilliVid's released full-resolution rollout checkpoint, run unmodified.

DFoT. Song et al. (2025) assigns an independent noise level to every token and denoises a sliding window under history guidance. We train it on the latents, clips and camera convention of Section A.2: 13 layers, width 1024, 8 heads and patch size 1 over the 8×8 latent grid, giving 249M parameters against the 250M budget of our own generator, on 101-frame latent clips. To achieve a fair comparison, we introduce two departures from the released code: (i) Camera conditioning is PRoPE (Li et al., 2025), which applies a parameter-free rotation of queries and keys, replacing the reference model's ray-encoding pathway, (ii) the noise-level schedule draws the context–horizon split per sample rather than at a fixed position, so every window the sampler later requests is in distribution, and History guidance remains unmodified and the model is optimized with AdamW (Loshchilov & Hutter, 2017) at $\mathrm{lr} = 1e{-}4$ at an effective batch size of 96 for $175{,}000$ steps in total. We use the reference sampler at 50 steps with stabilized history guidance at scale 4.0. Each window represents 50 latent frames and generates 51, so DFoT carries a bounded 50-frame history rather than the whole trajectory equivalent to ≈ 200 pixel frames which matches our training samples on average.

Table 5: **Memory capacity.** Reconstruction quality of the memory alone on *Memory Maze*, trained on 401-frame episodes. Doubling the memory parameters gains 0.3 dB at 401 frames and 1.3 dB at 3,201.

VARIANT	*401 frames*			*1601 frames*			*3201 frames*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SMART-SLOT MEMORY	32.7	0.940	0.024	32.4	0.936	0.024	31.9	0.931	0.026
SCALING UP MEMORY SLOTS (×2)	**33.0**	**0.943**	**0.023**	**33.3**	**0.947**	**0.020**	**33.2**	**0.946**	**0.020**

GeoCache. GeoCache is our reimplementation of the geometry-cached memory of Lyra 2.0 (Shen et al., 2026), strictly following its official repository: every observed frame is cached with its depth and camera, a read reprojects the cached geometry into the target camera to greedily retrieve the frames covering the most new target pixels, and use them with FramePack-style context (Zhang et al., 2025) both as latent slots and as depth-warped correspondence maps. We train it with the official codebase and recipe, including its self-augmentation against exposure bias, and change only what the maze setting requires.

The backbone is a pretrained 14-layer bidirectional model without cross-attention (292M parameters with Lyra's camera and correspondence encoders), as the released 14B system is untrainable at this scale, and each step attends an anchor, six temporal and nine spatial latent frames around a three-frame chunk, uncompressed to match our generator's window. For 64×64 frames the cache keeps full-resolution depth at every frame, and pixels without valid depth are excluded from retrieval. So that copying from the cache is learnable, training retrieves only from frames preceding the chunk, for three target views as at inference, without corrupting or dropping the retrieved slots; self-augmentation is widened to the full noise range with four denoising steps, since the released setting only presents near-clean history. The model is trained for about 78,000 iterations with a learning rate of $1e{-}4$, and a decaying learning rate schedule. We evaluate it with simulator depth, the most favourable setting for an explicit cache, starting the return leg from the last observed frame and a cache seeded with the whole forward leg and 35 denoising steps per chunk.

FramePack. Zhang et al. (2025) compresses history by representing older frames with progressively fewer tokens. We evaluate the released checkpoint of the reimplementation distributed with MilliVid (Chandratreya et al., 2026) to share its backbone, autoencoder, and sampler with other *Minecraft* arms. The progressive compression on subsequent 192, 48, 12, and the most recent 3 frame groups add up to 255 frames of history and occupies 3,840-token budget with 32-channel tokens. Three frames are emitted per step and is run at 50 sampling steps.

MilliVid. Chandratreya et al. (2026) conditions on a learned coarse-to-fine latent pyramid rather than a spatial repatchification, encoding each frame at four levels of 256, 64, 16 and 4 tokens and spending an equal token budget on each. Its history likewise spans 255 frames, and a chunk advances via 85 rollout steps per 192-frame chunk equivalent to $85 \times 50 = 4{,}250$ denoising steps. Both it and FramePack use the authors' released checkpoints with a DiT-B backbone (Peebles & Xie, 2023), the same size as our own *Minecraft* generator and no classifier-free guidance at test time.

Validating the state-parameter count. For our comparisons in Table 2, we keep a matched memory/context size for FramePack (Zhang et al., 2025), MilliVid (Chandratreya et al., 2026), and our approach for the long horizon memory.

FramePack and MilliVid. Uses 3,072 tokens for context (which include enough budget for [3, 12, 48, 192] frames at levels 0-3 respectively) and 768 tokens reserved for prediction in any given step. At inference time, the model sees a maximum of $3 + 12 + 48 + 192 = 255$ frames into the past. 3,072 tokens add up to $3{,}072 \times 32 = 98{,}304$ floats.

In SMART's 8 layer, 6 head per layer, 16 memory slot per head, 64 dimensional memory module, $2 \times 8 \times 6 \times 16 \times 64 = 98{,}304$ floats are essentially providing context for the diffusion process in an unbounded horizon. Although we validate that expanding the fixed number of memory slots also improves the reconstruction quality of our memory module (see Table 5), we use our 98,304 float memory module throughout this paper for fair comparison.

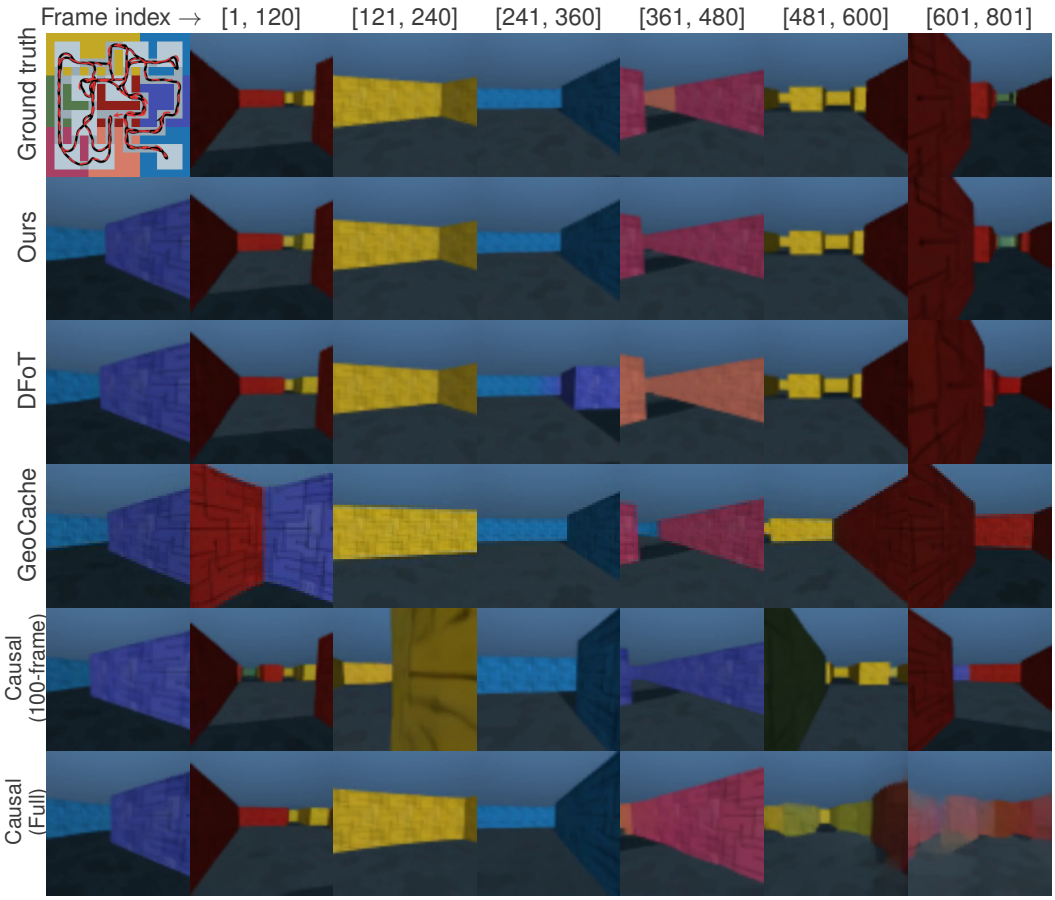


Figure 10: **Qualitative comparison on Memory Maze.** Generated frames for each method in Table 1, on the same trajectory with 823 frame context frame and 801 target frame generation.

A.6 Additional Experiments.

Qualitative results for Memory Maze. As reflected by the numbers in Table 1, SMART preserves the geometry, layout, and appearance of previously observed regions most consistently over the full return trajectory. In contrast, the baselines increasingly exhibit geometric drift, misplaced structures, and appearance changes as the revisit horizon grows, with the degradation becoming particularly visible in the later frame ranges.

Comparison with action-conditioned memory. Additionally, we demonstrate thay pose-conditioned memory can enable longer-horizon video generation with a comparison against StateSpaceDiffusers (Savov et al., 2026) which conditions its state-space memory on the agent's actions rather than querying it by camera pose. We reimplement it on *Memory Maze* inside our generator for the same number of training iterations for the SSM and the DiT. The DiT, chunk size, context window and draft conditioning match SMART, but the memory is a Mamba2 (Dao & Gu, 2024) recurrence driven by actions, and the DiT receives actions through its timestep modulation instead of the camera. Table 6 compares both on trajectories with forward legs of 25, 51, and 101 frames. Throughout the episodes, we observe that the action-conditioned memory degrades steadily with the horizon, losing 7.5 dB of PSNR on seen regions while SMART improves as more of the maze is observed. Because only regions observed during the forward leg are scored, this is a failure to recall rather than to invent, and we attribute it to the fact that the memory has no query: it predicts the next frame from the frames

Table 6: **Comparison against StateSpaceDiffusers (Savov et al., 2026) reimplementation on *Memory Maze*.** Both generators share the DiT architecture and draft conditioning and differ in how the memory is conditioned and read. Column groups are total trajectory length. PSNR and SSIM are computed over the pixels of each target frame that were visible during the forward leg and averaged per frame.

METHOD	$(25+21)+25=71$ *frames*			$(51+21)+51=123$ *frames*			$(101+21)+101=223$ *frames*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
STATESPACEDIFFUSERS	28.5	0.874	0.145	24.8	0.806	0.177	21.0	0.717	0.290
SMART	**30.2**	**0.906**	**0.108**	**32.3**	**0.931**	**0.073**	**33.3**	**0.939**	**0.039**

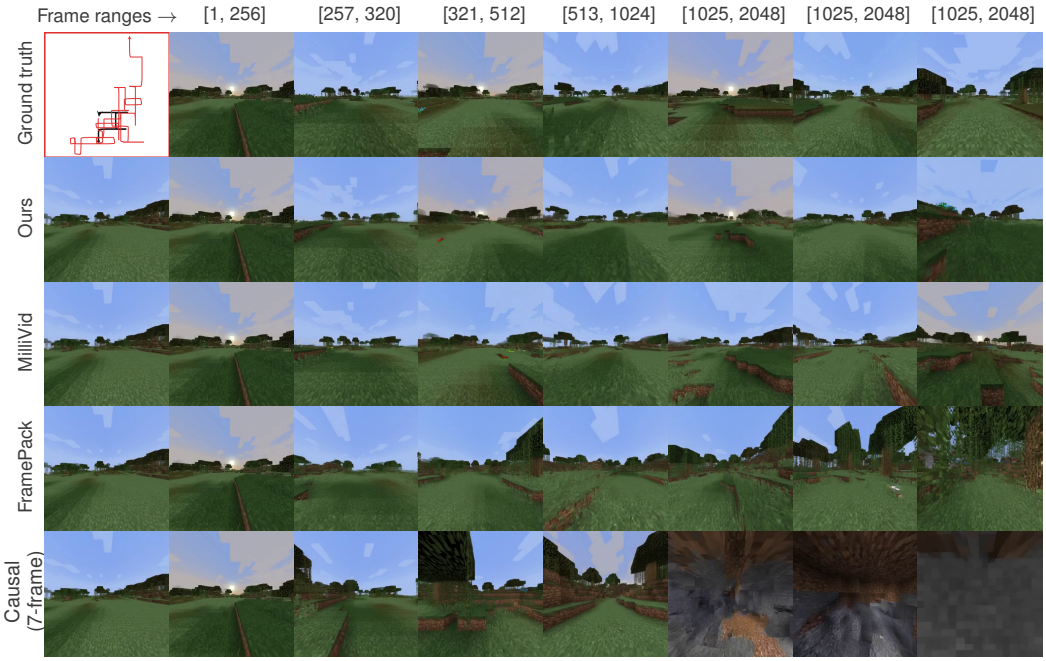


Figure 11: **Additional qualitative comparisons on *Minecraft*.** Generated frames for each method in Table 2, on the same trajectory, at the 2048 frame horizon with 256 initial context frames. Note that only SMART faithfully retrieves the position of the sun.

and actions so far. Recalling a place seen earlier requires the recurrence to integrate every action taken since that visit, and errors in this implicit reckoning grow with the length of the path.

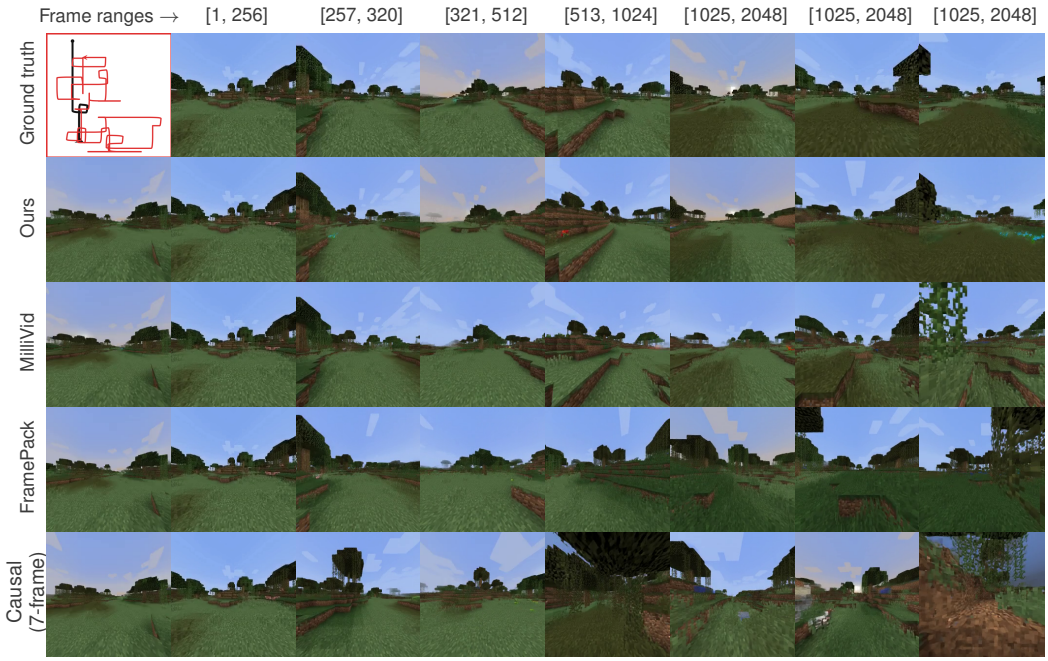


Figure 12: **Additional qualitative comparisons on *Minecraft*.**

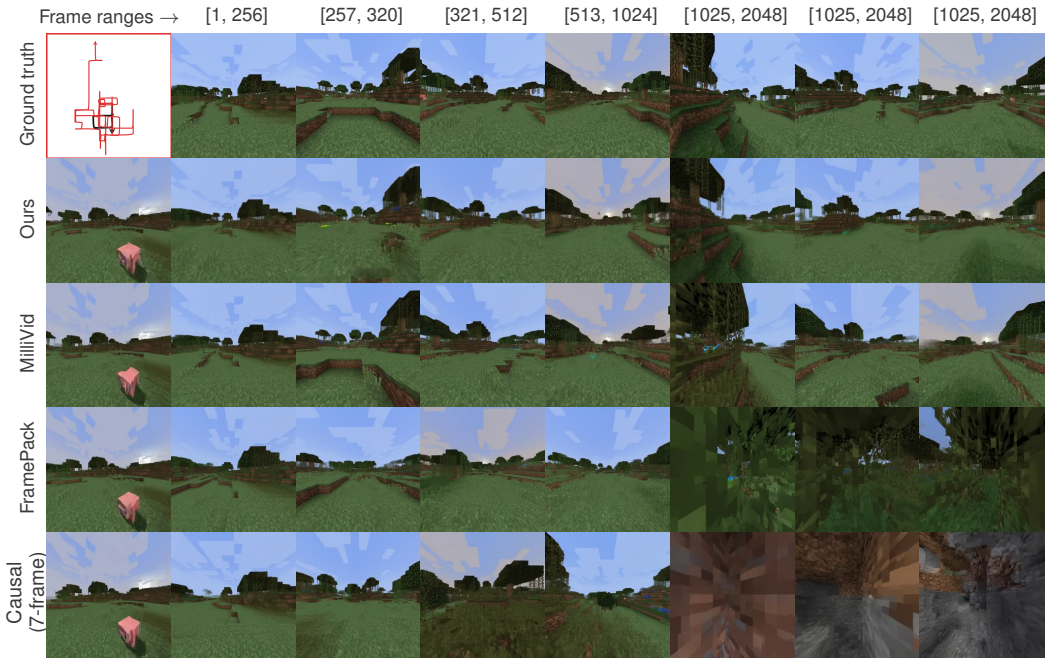


Figure 13: **Additional qualitative comparisons on *Minecraft*.**