# Robust Dreamer: Deviation-Aware Latent Gaussian Memory for Action-Controlled AR Video Generation

**Hanlin Chen**[1] **Jiaxin Wei**[2] **Xibin Song**[3] **Yifu Wang**[3]
**Steve Wang**[3] **Hongdong Li**[4] **Pan Ji**[3] **Gim Hee Lee**[1]
[1] School of Computing, National University of Singapore [2] Technische Universität München
[3] Vertex Lab [4] Australian National University
hanlin.chen@u.nus.edu gimhee.lee@nus.edu.sg

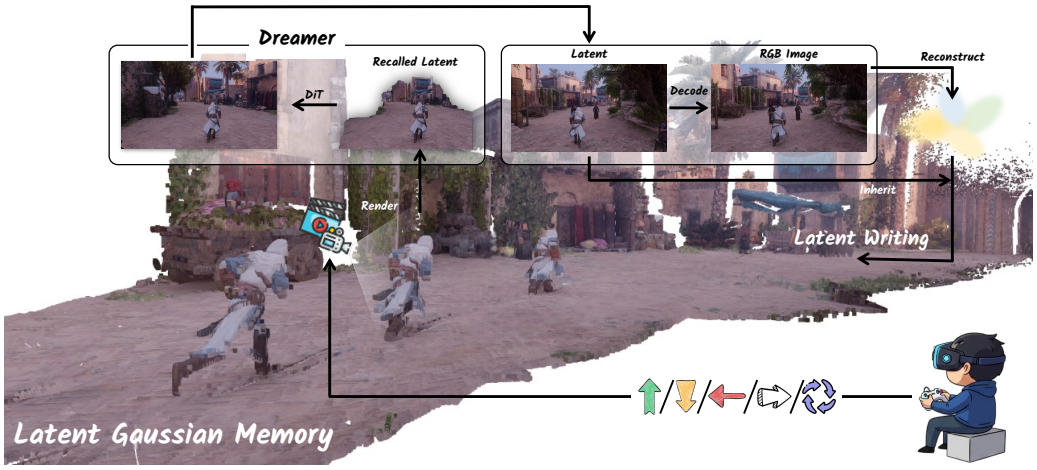


Figure 1: **Overview of the inference pipeline.** Our system performs long-horizon frame-by-frame generation through a closed-loop autoregressive process. First, a user action triggers memory recall via Gaussian Splatting, rendering a viewpoint-aligned latent. This latent conditions the proposed Dreamer to generate the next frame, which is subsequently decoded into RGB. Finally, the generated latent is directly inherited by the reconstructed primitives to update the memory.

## Abstract

Frame-wise action-controlled image-to-video generation is a promising paradigm for interactive world simulation, where each action conditions the next generation step. However, maintaining visual fidelity and 3D consistency over long autoregressive rollouts remains challenging. Existing 3D-aware methods often suffer from catastrophic drift due to two impediments: information loss from *Latent–RGB Cycling*, where generated latents are repeatedly decoded to RGB and re-encoded for future conditioning, and the training–inference gap induced by the *error-free hypothesis*, where clean training memory fails to match prediction-corrupted inference memory. To address these challenges, we present **Robust Dreamer**, a memory-augmented framework built around how to design 3D memory and how to use it robustly. First, we introduce **Latent Gaussian Memory**, which anchors diffusion latents inherited from the generation process to Gaussian primitives and recalls them via latent-space Gaussian splatting. This provides dense, geometry-aware, view-aligned conditioning while avoiding accumulated degradation from repeated VAE conversion. Second, we propose **Deviation Learning with Dy-**

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

**namic Deviation Archive**, which synthesizes rollout-induced latent deviations through a one-step approximation, stores them by autoregressive stage and denoising timestamp, and injects them into historical memory during training. This exposes the generator to realistic corrupted memory states and teaches internal correction before inference. Experiments on ScanNet, DL3DV, and OmniWorldGame demonstrate state-of-the-art long-horizon performance. Project page is available at https://hlinchen.github.io/projects/Robust_Dreamer/.

## 1 Introduction

World simulation has gained significant attention for its potential to model complex environments and predict the outcomes of actions. By rolling out plausible future trajectories, such models enable agents to plan, reason, and learn without expensive real-world interactions. Recent advances in video diffusion models have further propelled this field, offering high-fidelity frame synthesis and strong temporal coherence over short horizons [15, 55, 16]. However, extending these models from short clips to consistent and interactive long-horizon rollouts remains a formidable challenge. As the model recursively predicts future frames over long but finite horizons, errors accumulated over sequential steps often lead to severe degradation in visual quality and spatial consistency [40, 49].

Existing approaches to long-term autoregressive video generation generally fall into two categories. The first relies on 2D keyframe conditioning [36, 68, 76], employing attention or retrieval mechanisms to identify historical frames. While effective for stylistic coherence, these geometry-agnostic methods lack pixel-level supervision, often resulting in 3D inconsistencies and artifacts during significant viewpoint changes. The second category incorporates explicit 3D representations by lifting historical frames into a persistent 3D memory to condition generation [27, 67]. By organizing historical observations in a spatially grounded representation, 3D memory provides geometry-aware constraints for future views and offers a promising path toward more consistent long-horizon generation.

In this work, we aim to apply such 3D memory to *frame-wise action-controlled image-to-video generation*, where an initial image and step-wise controls drive an interactive rollout. This task is practical for embodied agents and interactive world simulation: each control signal conditions the next generation step, rather than committing one control to a long video chunk of dozens of frames. Here, "interactive" refers to step-wise controllability: each action can be selected after observing the current output. Our focus is robust per-action generation; real-time performance is outside the scope of this work. Here, 3D memory is especially important: every generated frame is written into a spatial memory and recalled to condition future actions from new viewpoints. However, we identify two key impediments in current 3D-aware frameworks. First, most methods store or render RGB observations before re-encoding them into the video model's latent space, causing *Latent–RGB Cycling* and discarding fine latent details. While a single VAE decode–encode cycle may be tolerable, frame-wise generation repeats this operation at every memory update and recall, amplifying small reconstruction errors into color drift and structural degradation, as shown in Fig. 2 (a). Second, current training often assumes clean ground-truth memory, whereas inference builds memory from imperfect model predictions with accumulated artifacts. This *error-free hypothesis* [28] creates a feedback loop: prediction deviations contaminate memory, corrupted memory biases future predictions, and the rollout gradually collapses, as shown in Fig. 2 (b).

To address these issues, we propose **Robust Dreamer**, a memory-augmented framework for frame-wise action-controlled image-to-video generation. Our solution is organized around how to design 3D memory and how to use it robustly. For memory design, inspired by latent-level manipulation in Wan-Move [12], we introduce **Latent Gaussian Memory**, which treats reconstruction as an incremental memory update in the diffusion latent space. Instead of decoding generated latents into an RGB proxy for storage, we inherit latent features directly from the generation process and anchor them to Gaussian primitives whose geometry is predicted by a feed-forward reconstruction module. During recall, Gaussian splatting aggregates these inherited latents with alpha-blending weights into the target viewpoint, producing a dense, pixel-aligned latent condition for the next frame. This keeps memory write and recall inside the high-dimensional latent space, preserves fine-grained 3D context, and avoids accumulated degradation from repeated VAE conversion. For memory usage, we introduce **Deviation Learning with Dynamic Deviation Archive**. Rather than assuming an idealized memory during training, we expose the generator to realistic corrupted historical contexts. To make this feasible without expensive autoregressive unrolling, we synthesize rollout-induced

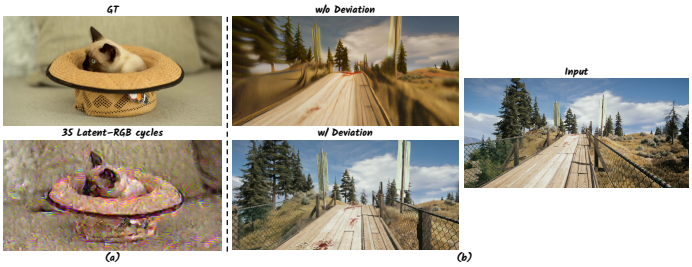


Figure 2: **Motivation.** (a) **Latent–RGB Cycling:** Repeatedly decoding a latent to RGB and encoding the RGB back to latent for 35 iterations causes catastrophic signal degradation and color distortion due to accumulated quantization errors. (b) **Deviation Learning:** The baseline (top), trained on clean memory, i.e., memory constructed from clean training frames/latents, suffers from structural collapse due to the training–inference gap. In contrast, our method (bottom) explicitly models accumulated errors, effectively mitigating drift and preserving geometric details.

latent deviations with a one-step approximation and store them in a **Dynamic Deviation Archive** hierarchically indexed by autoregressive stage and denoising timestamp. By sampling deviations from the archive and injecting them into historical memory states, the generator learns internal correction under the same type of imperfect memory it will encounter during autoregressive inference.

In summary, our contributions are:

- We propose **Latent Gaussian Memory**, a geometry-aware 3D memory that anchors inherited diffusion latents to Gaussian primitives and recalls them by splatting, enabling dense, view-aligned conditioning without repetitive latent–RGB cycling.
- We introduce **Deviation Learning with Dynamic Deviation Archive**, which synthesizes and stores realistic rollout-induced deviations to train the generator under corrupted memory states, bridging the training–inference gap and mitigating accumulated drift.
- We conduct comprehensive experiments on ScanNet, DL3DV, and OmniWorldGame, demonstrating state-of-the-art long-horizon performance, improved 3D consistency, and stronger robustness in frame-wise action-controlled video generation.

## 2 Related Work

**Camera-Controlled Video Generation.** Camera-controlled video generation conditions a video model on desired viewpoint changes. Early methods directly inject camera poses, motion vectors, or ray-based embeddings into diffusion backbones [63, 23, 1, 46, 37]. Recent action-controlled world models can also be viewed through this lens: keyboard or user actions are often translated into camera pose trajectories that drive interactive scene exploration [2, 35, 82, 24, 52, 29]. While pose-level conditioning improves controllability, numerical camera signals alone provide weak spatial constraints, especially under large viewpoint changes and revisits. To improve geometric fidelity, another line conditions generation on 3D-aware signals, such as depth-warped images, point-cloud renderings, or updatable spatial representations [78, 48, 77, 34, 83]. Our method follows this geometry-aware camera-control direction, but differs in two aspects: we generate frame by frame so that interaction can be injected at every step instead of committing to a long preplanned chunk, and we use the 3D representation as a persistent latent memory rather than an RGB proxy, together with deviation-aware training to make this memory robust during autoregressive rollout.

**Memory for Video Generation.** Autoregressive video generation requires mechanisms beyond a short sliding context. One line of work uses 2D memory, retrieving historical frames, visual tokens, or compressed context slots according to view overlap or temporal relevance [76, 68, 45, 25, 81]. Such memories are lightweight and compatible with pretrained video models, but they remain view-dependent and provide limited pixel-level geometric correspondence under large camera motion. Another line builds 3D memory with point clouds, surfels, spatial maps, or geometry-aware reconstruction to support revisit consistency [27, 66, 36, 67, 34, 42, 31, 83]. These methods are closer to our goal, yet most store or render RGB observations before re-encoding them into the generator, which accumulates latent–RGB cycling artifacts during autoregressive rollout. In contrast,

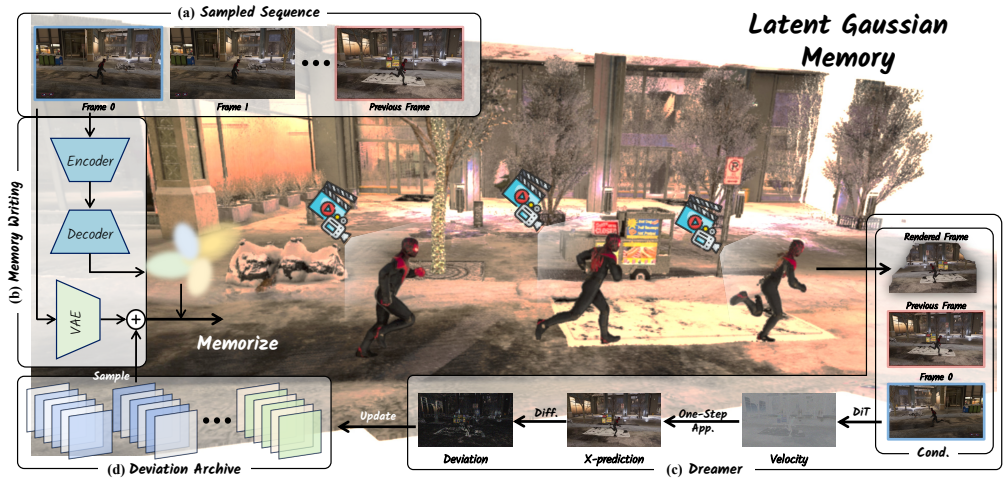


Figure 3: **Overview of the training pipeline.** (a) Variable-length subsequences provide historical context. (b) **Latent Gaussian Memory** is built from deviation-corrupted histories. (c) The Dreamer predicts velocity conditioned on clean anchor (frame 0), predecessor (previous frame), and recalled latent memory (rendered frame). (d) One-step deviations update the **Dynamic Deviation Archive**.

our Latent Gaussian Memory stores inherited diffusion latents on Gaussian primitives and recalls them by splatting, preserving dense 3D-aligned context without repeated VAE conversion.

**Train–Test Gap in Autoregressive Video Generation.** Autoregressive video diffusion enables streaming long-horizon generation [75, 54, 74], but training on ground-truth histories and testing on self-generated histories creates exposure bias and accumulated drift. The forcing family studies this mismatch by training or sampling under rollout-like conditions: Diffusion Forcing and history-guided diffusion model partially observed sequences [4, 51], Self-Forcing and Self-Forcing++ fine-tune on generated contexts [28, 13], while Rolling Forcing and self-resampling inject noisy or resampled histories to improve robustness [41, 22]. Causal Forcing further identifies an architectural gap in distilling bidirectional diffusion models into causal AR students and uses an AR teacher for ODE initialization to improve real-time interactive generation [86]. Stable Video Infinity studies error-bank recycling for very long rollouts [38], and recent analyses explicitly characterize error accumulation in AR video diffusion [57]. SVI and concurrent Matrix-Game 3.0 [64] recycle one-step prediction residuals. These approaches mostly address generic 2D histories or output frames. Our Deviation Learning targets the geometry-aware memory setting: it synthesizes realistic latent deviations and stores them in a Dynamic Deviation Archive, so the model learns to generate from corrupted memory states similar to those encountered at inference. Our method shares this residual estimator; its distinction is a stage-aware archive indexed jointly by denoising timestep and autoregressive depth, with deviations injected before 3D memory construction so that they propagate through Gaussian writing and view-aligned recall. This trains recovery from persistent 3D-memory corruption without autoregressive training rollouts, complementing the latent-native memory interface.

## 3 Robust Dreamer: Memory-Augmented Generation

In this work, we address action-controlled autoregressive video generation. Given an initial frame $\boldsymbol{I}_0$ and a step-wise control signal $c_i$, the model predicts the next frame $\boldsymbol{I}_{i+1}$ from the current rollout state. Our framework is built around two complementary memories. The **Latent Gaussian Memory** $\mathcal{M}$ provides geometrically grounded long-range context by storing generated content on Gaussian primitives, avoiding repeated latent–RGB conversion. The **Dynamic Deviation Archive** $\mathcal{A}$ models the errors accumulated by autoregressive rollout and exposes the generator to such corrupted memory during training. Sec. 3.1 gives the full training and inference pipeline, Sec. 3.2 details the latent Gaussian memory, and Sec. 3.3 presents Deviation Learning and the archive mechanism.

### 3.1 Memory-Conditioned Rollout Framework

We describe the overall generation and training protocol. We operate in the latent space of a pretrained video diffusion model and denote the latent of $\boldsymbol{I}_i$ as $\boldsymbol{X}_i$. The idea is to use a persistent latent memory to provide geometrically aligned long-range context during rollout, while training the generator with archived deviations that approximate inference-time errors.

**Memory-Conditioned Autoregressive Inference.** At inference time, Robust Dreamer maintains the Latent Gaussian Memory as an online state. The memory is initialized from the clean input pair $(\boldsymbol{I}_0, \boldsymbol{X}_0)$. At generation step $i$, the memory $\mathcal{M}_i$ contains the history up to frame $i$. Given the control $c_i$, we render $\mathcal{M}_i$ from the target viewpoint and obtain a view-aligned latent condition $\hat{\boldsymbol{X}}_{i+1}^{\mathcal{M}_i}$ for predicting $\boldsymbol{X}_{i+1}$. The Dreamer is conditioned on four complementary signals: the clean anchor $\boldsymbol{X}_0$, the latest rollout latent $\boldsymbol{X}_i$, the recalled memory feature $\hat{\boldsymbol{X}}_{i+1}^{\mathcal{M}_i}$, and the control $c_i$:

$$\boldsymbol{y}_i = \texttt{concat}(\boldsymbol{X}_0, \boldsymbol{X}_i, \hat{\boldsymbol{X}}_{i+1}^{\mathcal{M}_i}, c_i). \tag{1}$$

The recalled feature uses no future observation: it is rendered solely from $\mathcal{M}_i$ at the target viewpoint specified by $c_i$. The DiT denoiser predicts the next latent $\boldsymbol{X}_{i+1}$ from noise under this condition, and the latent is decoded into the RGB frame $\boldsymbol{I}_{i+1}$. The generated pair $(\boldsymbol{I}_{i+1}, \boldsymbol{X}_{i+1})$ is then written into the Latent Gaussian Memory to form $\mathcal{M}_{i+1}$, providing the geometric state for the subsequent control $c_{i+1}$.

**Deviation-Aware Training.** Training with clean ground-truth histories does not match the inference regime, where the memory is accumulated from imperfect predictions. To expose the generator to this rollout-induced distribution shift, we train the Dreamer with deviation-corrupted memory conditions. Given a sampled subsequence $\{\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{n+1}\}$, the first frame serves as the clean anchor and the final frame $\boldsymbol{X}_{n+1}$ is used as the clean target. We keep the anchor $\boldsymbol{X}_0$ uncorrupted, matching the practical setting where the initial frame is provided by the user, while subsequent frames are generated by the model and thus susceptible to accumulated drift. For each historical frame $1 \le i \le n$, we sample a deviation $\boldsymbol{D}_i$ from the Dynamic Archive $\mathcal{A}$ and inject it probabilistically:

$$\tilde{\boldsymbol{X}}_i = \boldsymbol{X}_i + \mathbb{I}_i \boldsymbol{D}_i, \quad \mathbb{I}_i \sim \text{Bernoulli}(p). \tag{2}$$

Unlike inference, where memory is updated online after each generated frame, training constructs a temporary corrupted memory $\tilde{\mathcal{M}}_n$ from the sampled history $(\boldsymbol{X}_0, \tilde{\boldsymbol{X}}_1, \ldots, \tilde{\boldsymbol{X}}_n)$. We render this memory for the target viewpoint to obtain $\hat{\boldsymbol{X}}_{n+1}^{\tilde{\mathcal{M}}_n}$ and form the training condition

$$\boldsymbol{y}_n = \texttt{concat}(\boldsymbol{X}_0, \tilde{\boldsymbol{X}}_n, \hat{\boldsymbol{X}}_{n+1}^{\tilde{\mathcal{M}}_n}, c_n). \tag{3}$$

The supervision remains the clean future latent $\boldsymbol{X}_{n+1}$. With a noisy latent $\boldsymbol{X}_{n+1}^t = t\boldsymbol{X}_{n+1} + (1-t)\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, we optimize the flow-matching objective to regress the velocity field

$$\mathcal{L} = \mathbb{E}_t \left[ \left\| \text{DiT}(\boldsymbol{X}_{n+1}^t, \boldsymbol{y}_n, t; \theta) - (\boldsymbol{X}_{n+1} - \boldsymbol{\epsilon}) \right\|_2^2 \right]. \tag{4}$$

We train the generator to denoise the future latent from deviation-corrupted history and memory recall. The archive $\mathcal{A}$ is updated online with one-step synthesized deviations, as detailed in Sec. 3.3.

### 3.2 Geometry-Grounded Latent Gaussian Memory

Action-controlled autoregressive video generation requires a memory that is both geometrically persistent and compatible with the latent space of the video generator. A common design in existing 3D-aware memories is to lift historical RGB frames into point-cloud-like 3D representations and later project them for future conditioning [27, 36, 67]. While this provides spatial grounding, the stored content repeatedly crosses the latent–RGB interface of the VAE, causing quantization and detail loss. Point-based projection is also sparse under novel views, leaving holes and disocclusion artifacts. Inspired by 3D Gaussian Splatting [30] and latent-space motion transfer [12], we instead store generated diffusion latents on continuous Gaussian primitives and recall them directly through alpha-composited splatting in latent space.

**Latent Gaussian Memory.** Formally, after processing frames up to frame $i$, we define the memory as a set of Gaussian primitives

$$\mathcal{M}_i = \{\mathbf{g}_s\}_{s=1}^{S_i}, \quad \mathbf{g}_s = (\boldsymbol{\mu}_s, o_s, \boldsymbol{q}_s, \boldsymbol{s}_s, \boldsymbol{x}_s), \tag{5}$$

where $\boldsymbol{\mu}_s \in \mathbb{R}^3$ is the center of the primitive in a global coordinate system, and $o_s$, $\boldsymbol{q}_s$, and $\boldsymbol{s}_s$ denote its opacity, rotation, and anisotropic scale, respectively. The last attribute $\boldsymbol{x}_s \in \mathbb{R}^C$ is the latent feature stored by the primitive. This parameterization decouples geometry from content: the Gaussian attributes determine where and how a memory element is projected, while $\boldsymbol{x}_s$ carries the semantic and appearance information inherited from the generated latent. For dynamic scenes, we further augment each primitive with time-dependent Gaussian attributes following 4D Gaussian representations [71], enabling a unified interface for static 3D recall and temporal 4D recall.

**Pixel-Aligned Latent Writing.** At generation step $i$, when the frame $\boldsymbol{I}_i$ becomes available, we write its latent $\boldsymbol{X}_i$ into the memory. We use a feed-forward reconstruction backbone initialized from CUT3R-style geometry prediction [58, 59, 32], to align the current observation with the accumulated global scene. Concretely, $\boldsymbol{I}_i$ is encoded into visual tokens using a ViT encoder [17] to obtain context-aware image tokens $\boldsymbol{F}'_i$ and a pose token $\boldsymbol{z}'_i$. On top of these context-aware tokens, we train DPT prediction heads to output the geometry and Gaussian attributes required by the memory bank:

$$\hat{\boldsymbol{P}}_i = \texttt{DPT}_{\text{global}}(\boldsymbol{F}'_i, \boldsymbol{z}'_i), \quad \{(\hat{o}_{ij}, \hat{\boldsymbol{q}}_{ij}, \hat{\boldsymbol{s}}_{ij})\}_{j=1}^{HW} = \texttt{DPT}_{\text{3DGS}}(\boldsymbol{F}'_i, \boldsymbol{z}'_i). \tag{6}$$

The global head follows the CUT3R-style world pointmap prediction and produces per-pixel 3D points $\hat{\boldsymbol{P}}_i$ in the global coordinate system, which serve as Gaussian centers $\boldsymbol{\mu}_{ij}$. The 3DGS head predicts the Gaussian parameters for each pixel-anchored primitive. To assign latent content, we first resize the generated latent map $\boldsymbol{X}_i$ to the image resolution and then read $\boldsymbol{x}_{ij}$ from the same pixel location as the corresponding primitive. Thus, memory writing lifts generated latents into globally aligned Gaussian primitives without predicting or re-encoding their semantic content.

For dynamic scenes, we use an additional 4DGS head to predict time-dependent rotation and scale for moving content. In practice, we also control the primitive count by downsampling the prediction resolution and merging nearby primitives through voxel-based pruning inspired by Scaffold-GS [43]. We provide these details in the supplementary material (Sec. A and Sec. B).

**Splatting-Based Latent Recall.** To condition the generation of frame $i+1$, we query $\mathcal{M}_i$ from the target viewpoint specified by the control $c_i$. Rather than rendering an RGB image and re-encoding it, we directly splat the latent features stored in $\mathcal{M}_i$ into the target view. For a pixel $k$ in the recalled latent map, let $\mathcal{N}(k)$ be the set of memory primitives whose projected Gaussians overlap that pixel, sorted by depth. The recalled feature is computed with the standard alpha-compositing form

$$\hat{\boldsymbol{X}}_{i+1}^{\mathcal{M}_i}(k) = \sum_{j \in \mathcal{N}(k)} T_j w_j \boldsymbol{x}_j, \quad T_j = \prod_{m<j} (1 - w_m), \tag{7}$$

where $w_j$ is the splatting weight of the $j$-th primitive at pixel $k$, and $T_j$ is the accumulated transmittance before that primitive. For static content, $w_j$ is determined by the projected 2D Gaussian density and opacity; for dynamic content, it is computed by slicing the corresponding 4D Gaussian at the target timestamp. We provide these retrieval details in Sec. B in the supplementary material. The resulting dense latent map $\hat{\boldsymbol{X}}_{i+1}^{\mathcal{M}_i}$ is spatially aligned with the requested viewpoint and is used as a structural condition for predicting $\boldsymbol{X}_{i+1}$, closing the write–recall loop of autoregressive generation.

### 3.3 Deviation Learning with Dynamic Deviation Archive

Autoregressive generation is vulnerable to accumulated deviation. During inference, each generated frame is written into the Latent Gaussian Memory and later recalled as part of the condition for future steps. Consequently, even small prediction errors are not isolated: they contaminate the memory, bias subsequent denoising, and are repeatedly reintroduced through the write–recall loop. This creates a training–inference mismatch, because standard training constructs memory from clean ground-truth histories, whereas inference constructs memory from the model's own imperfect outputs. A faithful solution would simulate full autoregressive rollouts during training, but doing so with iterative diffusion sampling is prohibitively expensive. We therefore introduce **Deviation Learning**: instead of replaying complete rollouts, we maintain a **Dynamic Deviation Archive** that stores realistic latent deviations synthesized online and reuses them to corrupt training histories.

**Archive Structure.** The archive is designed to approximate the non-stationary deviation distribution induced by the current generator. Deviations vary with the autoregressive stage, since errors at later frames contain longer accumulated history. They also depend on the denoising timestamp at which they are synthesized, because velocity prediction errors at different noise levels exhibit different

deviation patterns. We therefore organize the archive with two levels: the frame index $i$ of the autoregressive rollout and the denoising timestamp $t$:

$$\mathcal{A} = \{\mathcal{A}_i\}_{i=1}^{N}, \quad \mathcal{A}_i = \{\mathcal{A}_i^t\}_{t\in\mathcal{T}}, \quad \mathcal{A}_i^t = \{\boldsymbol{D}_{i,t,z}\}_{z=1}^{Z}. \tag{8}$$

Here $\mathcal{A}_i$ is the frame-specific archive for the $i$-th generated frame, $\mathcal{A}_i^t$ is the cell associated with denoising timestamp $t$, and $z$ indexes the $Z$ deviation slots stored in that cell. Each entry $\boldsymbol{D}_{i,t,z} \in \mathbb{R}^{C\times H\times W}$ stores a latent-space deviation between an approximate model prediction and the corresponding clean latent at frame $i$. This structure lets the archive preserve diverse error modes across both rollout depth and denoising state: early-stage entries capture local appearance or pose perturbations, whereas late-stage entries include accumulated color drift, geometric distortion, and semantic inconsistency.

**One-Step Deviation Synthesis.** A direct way to obtain deviations is to run the complete diffusion sampler and compare the generated latent with the ground truth. This is too expensive to perform for every training sample. Instead, we synthesize deviations with a one-step approximation under the current model. Given a noised latent $\boldsymbol{X}_i^t = t\boldsymbol{X}_i + (1-t)\boldsymbol{\epsilon}$ and the conditioning $\boldsymbol{y}_{i-1}$ used to predict $\boldsymbol{X}_i$, the DiT predicts a velocity

$$\hat{\boldsymbol{V}}_i^t = \mathrm{DiT}(\boldsymbol{X}_i^t, \boldsymbol{y}_{i-1}, t; \theta). \tag{9}$$

Following the flow-matching trajectory, we approximate the terminal latent with a single Euler step toward the data endpoint and define the synthesized deviation as

$$\bar{\boldsymbol{X}}_i = \boldsymbol{X}_i^t + (1-t)\hat{\boldsymbol{V}}_i^t, \quad \boldsymbol{D}_i = \bar{\boldsymbol{X}}_i - \boldsymbol{X}_i. \tag{10}$$

This computation is performed without gradient updates and serves only to estimate the deviation patterns induced by the current generator.

**Diversity-Preserving Refresh.** During training, each synthesized deviation is inserted into the archive cell matching its autoregressive frame and denoising timestamp. Thus, a deviation $\boldsymbol{D}_i$ synthesized for frame $i$ at timestamp $t$ enters $\mathcal{A}_i^t$. Each cell has capacity $Z$, bounding memory usage and preventing dominance by frequent stages or timestamps.

If the target cell is not full, the new deviation is appended. Once the cell reaches capacity, we update it with a diversity-preserving replacement rule. Rather than discarding a random entry, we replace the stored deviation most similar to the new one:

$$z^\star = \arg\min_z \|\boldsymbol{D}_i - \boldsymbol{D}_{i,t,z}\|_2, \quad \boldsymbol{D}_{i,t,z^\star} \leftarrow \boldsymbol{D}_i. \tag{11}$$

This rule removes redundant deviations while retaining rare modes such as structural warping, long-range appearance drift, or semantic inconsistency. Because the generator changes throughout training, the archive is continuously refreshed: newly synthesized deviations gradually replace stale deviations from earlier model states, allowing $\mathcal{A}$ to track the evolving error distribution.

When constructing corrupted latent Gaussian memory during training, we sample deviations according to each historical latent's stage. For a history frame at index $i$, we select $\mathcal{A}_i$, sample a timestamp $t \in \mathcal{T}$ uniformly, and draw an entry from $\mathcal{A}_i^t$. The sampled deviation is injected into the clean latent before memory writing, so both the predecessor condition and recalled memory inherit realistic inference-like artifacts. The archive is training-only: it calibrates the Dreamer to denoise under corrupted memory conditions, while inference maintains only the Latent Gaussian Memory and accumulates deviations naturally through generated frames autoregressively.

# 4 Experiments

We evaluate our method experimentally. We cover settings in Sec. 4.1, results in Sec. 4.2, and ablations in Sec. 4.3. More visualizations and analyses are in Appendix Sec. I and Sec. E, respectively.

## 4.1 Experimental Settings

**Datasets.** We evaluate our framework on three distinct datasets covering diverse scenarios, ranging from real-world indoor scenes to game-style simulation environments. First, we use ScanNet [14], a large-scale indoor dataset, to assess the model's ability to handle complex interior geometries and textures. Second, we employ DL3DV [39], which features high-quality 3D scenes with complex

Table 1: Short-/long-term (80/300 frames) generation results on the ScanNet.

| Metric | Short-term (80 frames) | | | | | | | Long-term (300 frames) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCtl | CCtl | WMem | WWarp | VMem | SForc | **Ours** | MCtl | CCtl | WMem | WWarp | VMem | SForc | **Ours** |
| PSNR ↑ | 12.14 | 12.35 | 14.23 | 15.59 | 13.38 | 13.95 | **16.89** | 9.14 | 9.35 | 11.12 | 11.76 | 9.21 | 9.38 | **13.43** |
| SSIM ↑ | 0.185 | 0.198 | 0.258 | 0.312 | 0.224 | 0.246 | **0.651** | 0.105 | 0.118 | 0.158 | 0.212 | 0.124 | 0.136 | **0.422** |
| LPIPS ↓ | 0.485 | 0.462 | 0.388 | **0.345** | 0.415 | 0.397 | 0.351 | 0.625 | 0.582 | 0.508 | 0.445 | 0.515 | 0.502 | **0.381** |
| FID ↓ | 85.34 | 81.12 | 42.50 | 38.65 | 56.40 | 44.20 | **16.82** | 125.34 | 118.12 | 72.50 | 68.65 | 96.40 | 92.40 | **38.42** |




Figure 4: **Qualitative results on ScanNet and DL3DV.** For each method, we visualize the first generated frame (left) and a later frame in the rollout (right). As generation progresses, baseline methods suffer from accumulated color drift and structural degradation, whereas our approach maintains consistent geometry and appearance without noticeable misalignment.

camera trajectories, to evaluate robustness in 3D-consistent view synthesis. Finally, to evaluate our method on long-horizon interactive trajectories, we utilize OmniWorldGame [85], a dataset designed for extended world-simulation rollouts.

**Evaluation metrics.** Following standard practices in video generation and novel view synthesis, we assess the performance quantitatively using four metrics covering pixel-level fidelity, perceptual quality, and distribution realism: (1) PSNR (Peak Signal-to-Noise Ratio) and (2) SSIM (Structural Similarity Index Measure) are used to evaluate pixel-wise accuracy and structural preservation compared to ground truth frames. (3) LPIPS (Learned Perceptual Image Patch Similarity) is employed to measure perceptual degradation, as it aligns better with human visual perception. (4) FID (Fréchet Inception Distance) is calculated to assess the overall visual realism and the distributional distance between the generated sequences and the real data.

### 4.2 Results

We present the main experimental results of our method, evaluating its performance in both short- and long-term video generation. We compare against a diverse set of state-of-the-art baselines to assess visual quality, temporal coherence, and robustness under long-horizon rollout. Specifically, we include MotionCtrl (MCtl) [63] and CameraCtrl (CCtl) [23] as representative non-autoregressive control-based methods, and Self Forcing (SForc) [28] as a recent frame-by-frame autoregressive approach designed to alleviate error accumulation. We further compare with memory-based autoregressive world modeling methods, WorldMem (WMem) [68], WorldWarp (WWarp) [31], and VMem [36], which enhance long-term consistency via explicit or implicit memory mechanisms.

**Comparisons on the ScanNet Dataset.** ScanNet provides long and continuous indoor video sequences with complex camera motion and rich geometric structure, making it particularly suitable for evaluating long-horizon video generation. Quantitative comparisons are reported in Tab. 1 for both short-term (80 frames) and long-term (300 frames) settings. As shown in the table, our method consistently achieves the best overall performance across all evaluation metrics in the long-term setting. In the short-term regime, our approach maintains competitive visual fidelity, while in the long-term setting it exhibits clear advantages, achieving higher PSNR and SSIM together with lower LPIPS and FID. This indicates stronger robustness to error accumulation during extended autoregressive rollout. Qualitative results in Fig. 4 further demonstrate that our method preserves coherent geometry and appearance over long sequences.

**Comparisons on the DL3DV Dataset.** DL3DV is a more challenging dataset that contains both indoor and outdoor scenes, with larger viewpoint changes and more diverse appearance variations.

Table 2: Quantitative comparison on the DL3DV and OmniWorldGame datasets.

| | DL3DV | | | | | | | OmniWorldGame | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Metric** | **MCtl** | **CCtl** | **SForc** | **WMem** | **WWarp** | **VMem** | **Ours** | **MCtl** | **CCtl** | **SForc** | **WMem** | **WWarp** | **VMem** | **Ours** |
| PSNR ↑ | 10.82 | 11.15 | 12.60 | 12.85 | 13.64 | 12.15 | **15.42** | 13.45 | 13.82 | 15.12 | 15.45 | 16.12 | 14.25 | **17.43** |
| SSIM ↑ | 0.145 | 0.162 | 0.215 | 0.224 | 0.285 | 0.194 | **0.522** | 0.245 | 0.258 | 0.352 | 0.315 | 0.382 | 0.284 | **0.541** |
| LPIPS ↓ | 0.524 | 0.495 | 0.418 | 0.412 | 0.385 | 0.435 | **0.291** | 0.385 | 0.364 | 0.295 | 0.345 | 0.312 | 0.354 | **0.242** |
| FID ↓ | 92.45 | 88.12 | 53.80 | 52.34 | 48.67 | 64.20 | **22.15** | 65.20 | 62.15 | 42.12 | 45.30 | 36.85 | 52.40 | **21.92** |

Table 3: Same-backbone attribution on ScanNet. All variants use Wan2.1-I2V-14B under an identical protocol.

| Method | **PSNR ↑** | **SSIM ↑** | **LPIPS ↓** | **FID ↓** |
|---|---|---|---|---|
| A. Wan2.1 AR | 13.13 | 0.240 | 0.419 | 53.69 |
| B. + SF Histories | 14.85 | 0.327 | 0.378 | 37.50 |
| C. + Latent Memory (clean) | 14.10 | 0.268 | 0.392 | 48.20 |
| D. + RGB Memory + Archive | 15.92 | 0.524 | 0.402 | 24.85 |
| E. + Latent Memory + Gaussian | 14.25 | 0.275 | 0.385 | 47.50 |
| F. Ours Full | **16.89** | **0.651** | **0.351** | **16.82** |

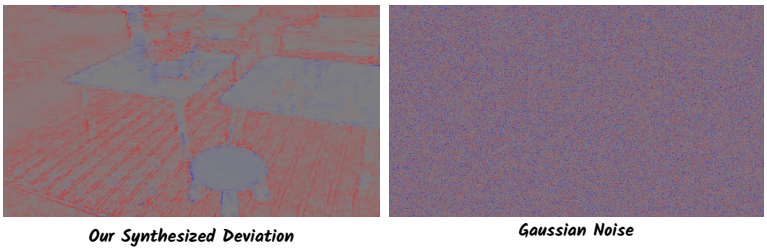


Figure 5: **Deviation patterns.** Comparison between our synthesized deviation (left) and Gaussian noise (right).

These factors introduce additional difficulty for autoregressive rollout. Quantitative results on DL3DV in Tab. 2 show that our method consistently outperforms all baselines across evaluation metrics. While baseline methods exhibit noticeable degradation when transitioning between indoor and outdoor environments, our approach maintains higher visual fidelity and temporal consistency. This demonstrates that the proposed method generalizes well across heterogeneous scene types and remains robust under more challenging real-world conditions.

**Comparisons on the OmniWorldGame Dataset.** OmniWorldGame features complex dynamic scenes with non-trivial object motion and interaction. Compared to static or quasi-static real-world datasets, OmniWorldGame poses additional challenges for maintaining temporal coherence in the presence of dynamic content. As shown in Tab. 2, our method achieves the best overall performance. Baseline methods tend to suffer from motion drift or temporal inconsistency as dynamics accumulate, whereas our approach preserves both scene structure and dynamic behavior over extended horizons. These results highlight the effectiveness of our method in handling dynamic environments.

Additional comparisons with Hunyuan World and Matrix-Game are provided in Appendix D.

### 4.3 Ablation Study

We conduct comprehensive ablation studies on the ScanNet dataset [14] to validate the effectiveness of our individual modules. The quantitative results are in Tab. 3. To separate the effects of the 14B generator, persistent memory, and deviation-aware training, all variants use the same Wan2.1-I2V-14B backbone, training data, action conditioning, LoRA configuration, resolution, and evaluation protocol.

**Backbone and Historical Context (Rows A–C).** Plain autoregressive fine-tuning of Wan2.1 (Row A) gives an FID of 53.69. Training on Self-Forcing-style generated histories without 3D memory improves FID to 37.50 (Row B), while adding clean Latent Gaussian Memory improves the plain AR baseline to 48.20 (Row C). These controlled variants show that the gains of our full model cannot be

attributed to the 14B backbone alone, and that persistent context without deviation-aware training still suffers from the train–test mismatch.

**Effectiveness of Latent Gaussian Memory (Rows D and F).** We first analyze the impact of performing memory writing and recall directly in the latent space with our Latent Gaussian Memory. We compare our full model (Row F) against a variant where the memory operates in the RGB pixel space (Row D, "+ RGB Memory + Archive"). In this setting, recalled content is decoded to RGB and then re-encoded via VAE for the next step. As shown in Row D, this *Latent–RGB Cycling* leads to a significant performance drop across all metrics due to repeated VAE reconstruction errors. This confirms that anchoring and inheriting features directly in the latent space is crucial for preserving high-frequency details.

**Impact of Deviation Learning (Rows C and F).** Next, we investigate the necessity of our Deviation Learning strategy in bridging the training–inference gap. The baseline trained solely on clean, ground-truth memory (Row C, "+ Latent Memory (clean)") exhibits poor performance. Without exposure to imperfect historical contexts during training, Model C lacks the capability to correct accumulated errors at inference time, leading to catastrophic drift. This result validates that explicitly training on error-corrupted memory states is essential for robust long-term generation.

**Validity of One-Step Deviation Synthesis (Rows E and F).** To verify whether our synthesized deviation captures realistic error patterns, we replace our one-step deviation synthesis with simple additive Gaussian noise (Row E, "+ Latent Memory + Gaussian"). The results in Row E show that training with simple noise yields suboptimal performance compared to our full method (Row F). This suggests that the distribution shift caused by autoregressive generation is structural and data-dependent, rather than random. Our proposed deviation synthesis successfully models these specific artifacts, enabling the model to learn effective internal correction. A visual comparison is provided in Fig. 2 (b). **Visualization of Deviation Patterns.** To further illustrate the difference between our synthesized deviation and random noise, we visualize the deviations decoded into RGB space in Fig. 5. Unlike standard Gaussian noise, which appears as uniform static, our synthesized deviation exhibits structure-aware patterns, such as edge blurring, that closely mirror the actual artifacts observed during inference.

## 5 Conclusion

In this paper, we presented **Robust Dreamer**, a memory-augmented framework for long-horizon world simulation that addresses the critical challenges of Latent–RGB Cycling and the training–inference gap. We introduced **Latent Gaussian Memory** to eliminate repeated latent–RGB conversion by anchoring inherited diffusion latents to Gaussian primitives and recalling them directly through latent-space splatting. We further proposed **Deviation Learning with Dynamic Deviation Archive** to robustify the Dreamer against accumulated drift by synthesizing rollout-induced latent deviations and injecting them into historical memory during training. Extensive experiments on ScanNet, DL3DV, and OmniWorldGame demonstrate that our approach significantly outperforms SOTA methods, maintaining high visual fidelity and strong 3D consistency over long horizons.

**Acknowledgment.** This research / project is supported by the National Research Foundation (NRF) Singapore, under its NRF-Investigatorship Programme (Award ID. NRF-NRFI09-0008), and the Tier 2 grant MOET2EP20124-0015 from the Singapore Ministry of Education.

## References


[1] Sherwin Bahmani, Ivan Skorokhodov, Guocheng Qian, Aliaksandr Siarohin, Willi Menapace, Andrea Tagliasacchi, David B Lindell, and Sergey Tulyakov. Ac3d: Analyzing and improving 3d camera control in video diffusion transformers. *arXiv preprint arXiv:2411.18673*, 2024.

[2] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

[3] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. PixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *CVPR*, 2024.

[4] 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, 2025.

[5] Hanlin Chen, Chen Li, Mengqi Guo, Zhiwen Yan, and Gim Hee Lee. Gnesf: Generalizable neural semantic fields. In *NeurIPS*, 2023.

[6] Hanlin Chen, Chen Li, Yunsong Wang, and Gim Hee Lee. NeuSG: Neural implicit surface reconstruction with 3D gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.

[7] Hanlin Chen, Fangyin Wei, and Gim Hee Lee. ChatSplat: 3D conversational gaussian splatting. *arXiv preprint arXiv:2412.00734*, 2024.

[8] Hanlin Chen, Fangyin Wei, Chen Li, Tianxin Huang, Yunsong Wang, and Gim Hee Lee. VCR-GauS: View consistent depth-normal regularizer for gaussian surface reconstruction. In *Advances in Neural Information Processing Systems*, volume 37, pages 139725–139750. Curran Associates, Inc., 2024.

[9] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025.

[10] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, 2024.

[11] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[12] Ruihang Chu, Yefei He, Zhekai Chen, Shiwei Zhang, Xiaogang Xu, Bin Xia, Dingdong Wang, Hongwei Yi, Xihui Liu, Hengshuang Zhao, et al. Wan-move: Motion-controllable video generation via latent trajectory guidance. *arXiv preprint arXiv:2512.08765*, 2025.

[13] Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation. *arXiv preprint arXiv:2510.02283*, 2025.

[14] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[15] Decart, Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, and Robert Wachen. Oasis: A universe in a transformer. 2024.

[16] Google DeepMind. Veo 3 technical report. Technical report, Google, 2025.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[18] Feilong Du, Yalan Zhang, Yihang Ji, Xiaokun Wang, Chao Yao, Jiri Kosinka, Steffen Frey, Alexandru Telea, and Xiaojuan Ban. GaussFluids: Reconstructing lagrangian fluid particles from videos via gaussian splatting. In *Pacific Graphics Conference Papers, Posters, and Demos*. The Eurographics Association, 2025.

[19] Y. Duan, S. Ren, J. Luo, Y. Chen, H. Wang, L. Zheng, and Q. Dai. 4d radiance fields with multi-scale occupancy networks for dynamic scene reconstruction. In *CVPR*, 2024.

[20] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

[21] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. MD-Splatting: Learning metric deformation from 4D gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583v1*, 2023.

[22] Yuwei Guo, Ceyuan Yang, Hao He, Yang Zhao, Meng Wei, Zhenheng Yang, Weilin Huang, and Dahua Lin. End-to-end training for autoregressive video diffusion via self-resampling. *arXiv preprint arXiv:2512.15702*, 2025.

[23] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. In *ICLR*, 2025.

[24] 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.

[25] Yicong Hong, Yiqun Mei, Chongjian Ge, Yiran Xu, Yang Zhou, Sai Bi, Yannick Hold-Geoffroy, Mike Roberts, Matthew Fisher, Eli Shechtman, et al. RELIC: Interactive Video World Model with Long-Horizon Memory. *arXiv preprint arXiv:2512.04040*, 2025.

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. ACM, 2024.

[27] Tianyu Huang, Wangguandong Zheng, Tengfei Wang, Yuhao Liu, Zhenwei Wang, Junta Wu, Jie Jiang, Hui Li, Rynson Lau, Wangmeng Zuo, et al. Voyager: Long-range and world-consistent video diffusion for explorable 3d scene generation. *ACM Transactions on Graphics (TOG)*, 44(6):1–15, 2025.

[28] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

[29] InSpatio Team. InSpatio: A Real-Time 4D World Simulator via Spatiotemporal Autoregressive Modeling. *arXiv preprint arXiv:2604.07209*, 2026.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023.

[31] Hanyang Kong, Xingyi Yang, Xiaoxu Zheng, and Xinchao Wang. Worldwarp: Propagating 3d geometry with asynchronous video diffusion. *arXiv preprint arXiv:2512.19678*, 2025.

[32] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r. *arXiv:2406.09756*, 2024.

[33] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024.

[34] Guangyuan Li, Siming Zheng, Shuolin Xu, Jinwei Chen, Bo Li, Xiaobin Hu, Lei Zhao, and Peng-Tao Jiang. Magicworld: Interactive geometry-driven video world exploration. *arXiv preprint arXiv:2511.18886*, 2025.

[35] Jiaqi Li, Junshu Tang, Zhiyong Xu, Longhuang Wu, Yuan Zhou, Shuai Shao, Tianbao Yu, Zhiguo Cao, and Qinglin Lu. Hunyuan-gamecraft: High-dynamic interactive game video generation with hybrid history condition. *arXiv preprint arXiv:2506.17201*, 2025.

[36] Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. Vmem: Consistent interactive video scene generation with surfel-indexed view memory. *arXiv preprint arXiv:2506.18903*, 2025.

[37] Teng Li, Guangcong Zheng, Rui Jiang, Tao Wu, Yehao Lu, Yining Lin, Xi Li, et al. Realcam-i2v: Real-world image-to-video generation with interactive complex camera control. *arXiv preprint arXiv:2502.10059*, 2025.

[38] Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable video infinity: Infinite-length video generation with error recycling. *arXiv preprint arXiv:2510.09212*, 2025.

[39] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. DL3DV-10K: a large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024.

[40] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14458–14467, 2021.

[41] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025.

[42] Dongyue Lu, Ao Liang, Tianxin Huang, Xiao Fu, Yuyang Zhao, Baorui Ma, Liang Pan, Wei Yin, Lingdong Kong, Wei Tsang Ooi, and Ziwei Liu. See4d: Pose-free 4d generation via auto-regressive video inpainting. *arXiv preprint arXiv:2510.26796*, 2025.

[43] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.

[44] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: tracking by persistent dynamic view synthesis. In *3DV*, 2024.

[45] Yuta Oshima, Yusuke Iwasawa, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Worldpack: Compressed memory improves spatial consistency in video world modeling. *arXiv preprint arXiv:2512.02473*, 2025.

[46] Stefan Popov, Amit Raj, Michael Krainin, Yuanzhen Li, William T Freeman, and Michael Rubinstein. Camctrl3d: Single-image scene exploration with precise 3d camera control. *arXiv preprint arXiv:2501.06006*, 2025.

[47] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024.

[48] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[49] Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. Genwarp: Single image to novel views with semantic-preserving generative warping. In *NeurIPS*, 2024.

[50] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024.

[51] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion, 2025.

[52] 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.

[53] Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction. *arXiv preprint arXiv:2410.06245*, 2024.

[54] Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

[55] 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.

[56] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[57] Jing Wang et al. Error analyses of auto-regressive video diffusion models: A unified framework. *arXiv preprint arXiv:2503.10704*, 2025.

[58] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025.

[59] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024.

[60] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: geometric 3D vision made easy. In *CVPR*, 2024.

[61] Yunsong Wang, Hanlin Chen, and Gim Hee Lee. Gov-nesf: Generalizable open-vocabulary neural semantic fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20443–20453, 2024.

[62] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat: Generalizable 3d gaussian splatting towards free-view synthesis of indoor scenes. *arXiv preprint arXiv:2405.17958*, 2024.

[63] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[64] Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 3.0: Real-Time and Streaming Interactive World Model with Long-Horizon Memory. *arXiv preprint arXiv:2604.08995*, 2026.

[65] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and W. Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[66] Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. *arXiv preprint arXiv:2507.07982*, 2025.

[67] Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. *arXiv preprint arXiv:2506.05284*, 2025.

[68] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory. *arXiv preprint arXiv:2504.12369*, 2025.

[69] Weixing Xie, Junfeng Yao, Xianpeng Cao, Qiqin Lin, Zerui Tang, Xiao Dong, and Xiaohu Guo. SurgicalGaussian: Deformable 3D gaussians for high-fidelity surgical scene reconstruction. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume 15006 of *Lecture Notes in Computer Science*, pages 617–627. Springer Nature Switzerland, 2024.

[70] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21924–21935, 2025.

[71] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023.

[72] Botao Ye, Sifei Liu, Haofei Xu, Li Xueting, Marc Pollefeys, Ming-Hsuan Yang, and Peng Songyou. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.

[73] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

[74] Hidir Yesiltepe, Tuna Han Salih Meral, Adil Kaan Akan, Kaan Oktay, and Pinar Yanardag. Infinity-rope: Action-controllable infinite video generation emerges from autoregressive self-rollout. *arXiv preprint arXiv:2511.20649*, 2025.

[75] Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22963–22974, 2025.

[76] Jiwen Yu, Jianhong Bai, Yiran Qin, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Context as memory: Scene-consistent interactive long video generation with memory retrieval. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–11, 2025.

[77] Mark YU, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. *arXiv preprint arXiv:2503.05638*, 2025.

[78] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. ViewCrafter: taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024.

[79] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024.

[80] Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. *arXiv preprint arXiv:2408.13770*, 2024.

[81] 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 *NeurIPS*, 2025.

[82] Yifan Zhang, Chunli Peng, Boyang Wang, Puyi Wang, Qingcheng Zhu, Fei Kang, Biao Jiang, Zedong Gao, Eric Li, Yang Liu, et al. Matrix-game: Interactive world foundation model. *arXiv preprint arXiv:2506.18701*, 2025.

[83] Jinjing Zhao, Fangyun Wei, Zhening Liu, Hongyang Zhang, Chang Xu, and Yan Lu. Spatia: Video Generation with Updatable Spatial Memory. *arXiv preprint arXiv:2512.15716*, 2025.

[84] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19680–19690, 2024.

[85] Yang Zhou, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Haoyu Guo, Zizun Li, Kaijing Ma, Xinyue Li, Yating Wang, Haoyi Zhu, Mingyu Liu, Dingning Liu, Jiange Yang, Zhoujie Fu, Junyi Chen, Chunhua Shen, Jiangmiao Pang, Kaipeng Zhang, and Tong He. Omniworld: A multi-domain and multi-modal dataset for 4d world modeling. *arXiv preprint arXiv:2509.12201*, 2025.

[86] 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.

[87] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

# A Memory Compression Strategy.

Standard feed-forward 3DGS methods typically assign one Gaussian per pixel. While feasible for sparse-view settings, this approach becomes intractable for our frame-by-frame autoregressive generation. For instance, a standard 81-frame sequence at $512 \times 288$ resolution would yield over 11 million Gaussians, creating an excessive burden on both memory storage and retrieval efficiency. To maintain a compact and efficient memory representation, we employ a dual-level pruning strategy.

First, at the 2D level, we recognize that per-pixel assignment leads to spatial redundancy; thus, we apply a $2\times$ downsampling to the prediction resolution, reducing the initial primitive count by $75\%$. Second, at the 3D level, since autoregressive generation often reconstructs overlapping geometric surfaces across adjacent frames, simply accumulating primitives results in significant redundancy. To resolve this, we introduce a Voxel-based Pruning mechanism inspired by [43] to fuse spatially adjacent primitives into a unified representation.

Specifically, we discretize the spatial domain into a grid with voxel size $\epsilon$. For each of the $N$ generated Gaussians with centers $\{\boldsymbol{\mu}_j\}_{j=1}^{N}$, we determine its corresponding voxel index $\boldsymbol{v}_j$: $\boldsymbol{v}_j = \left\lfloor \frac{\boldsymbol{\mu}_j}{\epsilon} \right\rceil$. Let $\mathcal{G}_s = \{j \mid \boldsymbol{v}_j = s\}$ denote the set of Gaussians falling into voxel $s$. To ensure the pruning process remains end-to-end differentiable, we predict a confidence score $c_j$ for each Gaussian and compute intra-voxel contribution weights $m_j$ via a local softmax: $m_j = \frac{\exp(c_j)}{\sum_{k\in\mathcal{G}_s}\exp(c_k)}$, for $j \in \mathcal{G}_s$. Finally, all Gaussian attributes $a_j$, including opacity, scale, rotation, and crucially, the inherited latent feature $\boldsymbol{x}_j$, are aggregated into a single representative voxel attribute $\bar{a}_s$ via weighted summation: $\bar{a}_s = \sum_{j\in\mathcal{G}_s} m_j\, a_j$. The final memory bank is thus parameterized by the aggregated voxel attributes $\{(\bar{\boldsymbol{\mu}}_s, \bar{o}_s, \bar{\boldsymbol{q}}_s, \bar{\boldsymbol{s}}_s, \bar{\boldsymbol{x}}_s)\}_{s=1}^{S}$. This strategy dramatically reduces the primitive count while preserving the gradient flow, ensuring that our incremental memory updates remain both geometrically accurate and computationally scalable. For notational simplicity, we omit the bar accent $(\bar{\cdot})$ in the remainder of the paper and refer to these aggregated voxel attributes simply as Gaussian parameters.

# B Details of Static and Temporal Memory Retrieval

**Static Memory Retrieval.** For static background elements, we employ standard 3D Gaussian Splatting. Each Gaussian is parameterized by opacity $\alpha$, covariance $\boldsymbol{\Sigma}$, center $\boldsymbol{\mu}$, and the latent feature $\boldsymbol{x}$. The covariance is decomposed into scaling $\boldsymbol{S}$ and rotation $\boldsymbol{R}$ as $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top$. Given the viewing transformation $\boldsymbol{W}_{i+1}$ and Jacobian $\boldsymbol{J}$, the covariance is transformed to $\boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{W}_{i+1}\boldsymbol{\Sigma}\boldsymbol{W}_{i+1}^\top\boldsymbol{J}^\top$. The blending weight for static elements is determined by the 2D projected spatial density:

$$w_j = G(\boldsymbol{p}_j)\alpha_j, \tag{12}$$

where $G(\boldsymbol{p}) = \exp\left(-\frac{1}{2}(\boldsymbol{p}-\boldsymbol{\mu}')^\top{\boldsymbol{\Sigma}'}^{-1}(\boldsymbol{p}-\boldsymbol{\mu}')\right)$ and $\boldsymbol{\mu}'$ is the projected 2D mean.

**Temporal Memory Retrieval.** To recall dynamic content consistent with the scene's evolution, we employ 4D Gaussian Splatting, predicting the object state at timestamp $t_{i+1}$. We extend the primitives with temporal quaternion $\boldsymbol{q}_{j,t}$ and scale $s_{j,t}$ to form 4D Gaussians centered at $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_z, \mu_t)$. The 4D covariance utilizes a diagonal scaling $\boldsymbol{S}$ and a 4D rotation matrix $\boldsymbol{R}$ constructed from left/right unit quaternions $\boldsymbol{q}_l = [s_l, \boldsymbol{v}_l^\top]^\top$ and $\boldsymbol{q}_r = [s_r, \boldsymbol{v}_r^\top]^\top$:

$$\boldsymbol{R} = \begin{bmatrix} s_l & -\boldsymbol{v}_l^\top \\ \boldsymbol{v}_l & s_l\mathbf{I} + [\boldsymbol{v}_l]_\times \end{bmatrix} \begin{bmatrix} s_r & -\boldsymbol{v}_r^\top \\ \boldsymbol{v}_r & s_r\mathbf{I} - [\boldsymbol{v}_r]_\times \end{bmatrix}, \tag{13}$$

where $s_{\{\cdot\}} \in \mathbb{R}$ and $\boldsymbol{v}_{\{\cdot\}} \in \mathbb{R}^3$ denote the scalar and vector parts of the quaternions, respectively. The notation $[\cdot]_\times$ represents the $3\times 3$ skew-symmetric matrix operator equivalent to the cross product, and $\mathbf{I}$ is the identity matrix. The recall process follows the slicing approach [71]. The weight $w_j$ for dynamic elements incorporates both the conditional spatial distribution and the marginal temporal probability:

$$w_j = G(\boldsymbol{x}_j|t_{i+1})G(t_{i+1})\alpha_j. \tag{14}$$

The marginal probability is $G(t) = \mathcal{N}(t; \mu_t, \boldsymbol{\Sigma}_{4,4})$, and the conditional spatial term is defined as $G(\boldsymbol{x}|t) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{x}|t})^\top\boldsymbol{\Sigma}_{\boldsymbol{x}|t}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{x}|t})\right)$. The conditional mean and covariance are derived from the multivariate Gaussian properties: $\boldsymbol{\mu}_{\boldsymbol{x}|t} = \boldsymbol{\mu}_{1:3} + \boldsymbol{\Sigma}_{1:3,4}\boldsymbol{\Sigma}_{4,4}^{-1}(t_{i+1}-\mu_t)$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}|t} = \boldsymbol{\Sigma}_{1:3,1:3} - \boldsymbol{\Sigma}_{1:3,4}\boldsymbol{\Sigma}_{4,4}^{-1}\boldsymbol{\Sigma}_{4,1:3}$.

# C Implementation Details

Our *Robust Dreamer* is implemented in PyTorch, utilizing the CUDA-optimized `gsplat` library [73] for 3D Gaussian Splatting (3DGS) rendering. For the Pixel-Aligned Latent Writing module in Latent Gaussian Memory

(Sec. 3.2), we employ a ViT-Large [17] encoder and a ViT-Base decoder, equipped with DPT prediction heads. This module is initialized with pretrained weights from CUT3R [58] to facilitate accurate initial point cloud estimation. Our training pipeline proceeds in two stages: Latent Gaussian Memory construction and autoregressive video diffusion. In the first stage, we freeze the backbone and exclusively optimize the Gaussian and global heads. We sample video subsequences ranging from 5 to 81 frames to predict explicit Gaussian representations, using subsequent frames for supervision. All input images are resized to $512 \times 288$. In the second stage, we build our *Robust Dreamer* upon the Wan2.1-I2V-14B-480P model [55]. To ensure deployment flexibility, we employ Low-Rank Adaptation (LoRA) for fine-tuning; this design allows users to seamlessly inject our method into private or customized models. Detailed hyperparameters are provided in Tab. 4. All experiments were conducted on a cluster of 8 NVIDIA H200 GPUs, each equipped with 144 GB of memory. The reconstruction training in the first stage and the diffusion training in the second stage each take approximately 31 hours under this setup. During inference, generating an 81-frame video requires about 14 minutes, which is comparable to other Wan2.1 models at the 14B scale.

**Action representation and comparison protocol.** Across ScanNet, DL3DV, and OmniWorldGame, the action condition is a camera pose during both training and inference. We use the first camera as the reference coordinate system and express every subsequent pose in this coordinate system. At interactive inference, the target pose is externally specified rather than estimated by the reconstruction module. For the end-to-end comparisons, we fine-tune MotionCtrl, CameraCtrl, Self Forcing, and WorldMem on the same data with their official training pipelines. Because complete training pipelines for WorldWarp and VMem are unavailable, we use their official checkpoints and inference settings. All methods are evaluated at the same resolution and under the same rollout protocol while retaining their native backbones. Therefore, the main end-to-end tables measure practical system-level performance and may also reflect differences in backbone and original training data; the same-backbone study in Tab. 3 isolates the contributions of our memory and deviation mechanisms.

Table 4: **Hyperparameter settings for DiT training**. Parameters are categorized into optimization strategies, model architecture, and deviation learning configurations.

| Parameter | Value | Description |
|---|---|---|
| ***Optimization & Training Strategy*** | | |
| Learning rate | $2.0 \times 10^{-5}$ | Learning rate for Adam optimizer |
| Max epochs | 20 | Maximum number of training epochs |
| Gradient clipping | 1.00 | Maximum norm for gradient clipping |
| Gradient accumulation | 1 | Number of steps for gradient accumulation |
| Training strategy | deepspeed_stage_2 | Distributed training strategy (DeepSpeed) |
| Data workers | 1 | Number of workers for data loading |
| Gradient checkpointing | Yes | Activation checkpointing for memory efficiency |
| Checkpointing offload | No | Offload checkpoint activations to CPU |
| ***Model Architecture & LoRA Configuration*** | | |
| Architecture | LoRA | Type of fine-tuning architecture |
| LoRA rank ($r$) | 128 | Rank dimension for Low-Rank Adaptation |
| LoRA alpha ($\alpha$) | 128 | Scaling factor for LoRA |
| LoRA init | Kaiming | Initialization scheme for LoRA weights |
| LoRA position | q, k, v, o, ffn.0, ffn.2 | Transformer modules applying LoRA |
| Frame resolution | $480 \times 832$ | Spatial resolution ($H \times W$) of frames |
| Video frames | 5–81 | Range of sequence length for training samples |
| ***Deviation Learning & Dynamic Deviation Archive*** | | |
| Warmup iterations | 20 | Warmup steps for deviation gathering |
| Deviation $p$ | 0.9 | Probability of deviation injection |
| Clean input $p$ | 0.5 | Probability of conditioning on clean history |
| Timestamp grids | 50 | Number of denoising timestamp bins in Dynamic Deviation Archive |
| Archive capacity $Z$ | 500 | Capacity of each archive cell in Dynamic Deviation Archive |

## D Comparisons with Recent World Models

We compare with Hunyuan World and Matrix-Game on OmniWorldGame, adapting their next-chunk outputs to our next-latent, per-action setting. Our model generates one temporal latent (four RGB frames) per action. To match this output granularity, only the first temporal latent of each chunk generated by each baseline is retained for both output and history update. The table lists the native generation formulation, whereas all reported metrics use this adapted protocol.

Table 5: Comparison with recent world models on OmniWorldGame under an adapted per-action protocol. For the next-chunk baselines, only the first temporal latent of each generated chunk is retained for output and history update.

| Method | Native formulation | PSNR ↑ | SSIM ↑ | LPIPS ↓ | FID ↓ |
|---|---|---|---|---|---|
| Hunyuan World | Next-chunk | 15.24 | 0.372 | 0.352 | 40.86 |
| Matrix-Game | Next-chunk | 16.35 | 0.421 | 0.291 | 34.21 |
| **Ours** | Next-latent | **17.43** | **0.541** | **0.242** | **21.92** |

Under this adapted protocol, our method achieves the best PSNR, SSIM, LPIPS, and FID in Tab. 5. These results characterize performance under the per-action protocol rather than the baselines' native chunk-generation setting.

## E Runtime, Memory, and Robustness Analysis

**Runtime and memory costs.** *Robust Dreamer* is efficient in both GPU memory and runtime, introducing only negligible overhead beyond the base video diffusion model. The Dynamic Deviation Archive is stored on CPU and occupies about 22 GB of CPU memory. During training, we transfer only the sampled deviation from CPU to GPU, which costs about 0.6 MB for each frame. Each one-step deviation approximation takes about 0.266 ms, since it performs only one local approximation, reuses existing forward-pass activations, and is computed in a `no_grad` context.

For a standard 81-frame sequence at $512 \times 288$ resolution, the Latent Gaussian Memory uses only about 100 MB of GPU memory, with an average runtime of 76.34 ms per frame. Compared with prior 3D-aware memory methods such as VMem [36] and WorldWarp [31], this is substantially more efficient. A standard per-pixel 3D-aware memory implementation would otherwise produce over 11 million Gaussians and require roughly 500 MB of GPU memory. Our efficiency comes from the dual-level pruning strategy in Sec. A, which prunes the 3D representation from both the 2D and 3D perspectives. Overall, this reduces memory usage by about 81% and runtime by about 18%. The detailed cost breakdown is provided in Tab. 6. Across datasets, the footprint is governed primarily by the unique scene space covered by the trajectory and the resulting number of occupied voxels after voxel-level merging, rather than by trajectory length alone. Exploring a new region introduces new occupied voxels and grows the memory, whereas revisiting previously covered regions largely reuses existing voxels and therefore adds little memory. DL3DV can consequently require slightly more memory when its trajectories cover broader or more geometrically complex scenes, but this behavior is not specific to the dataset itself.

**Cost of maintaining the Dynamic Deviation Archive.** Maintaining and updating the Dynamic Deviation Archive introduces very small overhead in practice. In our implementation, archive maintenance costs at most about 350 ms per training iteration, while one full training iteration takes about 11.1 s. This corresponds to only about 3.2% overhead in the worst case. Importantly, when training the current frame, we do not run an additional forward pass over historical frames. We simply sample stored deviations from the archive and inject them into historical latents before memory construction, as described in Sec. 3.3. Since the archive is implemented as a dictionary indexed by autoregressive stage and denoising timestamp, sampling and lookup are lightweight and their cost is negligible compared with the diffusion training step.

**Efficiency and role of memory recall.** Our memory recall mechanism directly renders the stored Gaussians into the target view through Gaussian Splatting (Sec. B and Eq. 7). This operation is efficient in practice: real-time 3DGS systems and optimized splatting libraries commonly report rendering speeds above 100 FPS [30, 73], so the recall step adds only modest overhead relative to the 14B diffusion backbone. Although adjacent frames are often visually similar, small viewpoint changes can still reveal newly visible or previously unobserved regions, and those regions may have been better captured from earlier frames. The recall module retrieves such historical information as a view-aligned condition, helping the generated frame remain consistent with the accumulated history. This is consistent with the broader use of memory in autoregressive video generation and world modeling, including Self-Forcing [28], WorldMem [68], WorldWarp [31], and VMem [36].

Table 6: **Runtime and memory costs.** We report costs for a standard 81-frame sequence at $512 \times 288$ resolution.

| Method / Component | GPU Memory | CPU Memory | Runtime | Notes |
|---|---|---|---|---|
| Standard 3D-aware memory | ∼500 MB | – | ∼93 ms / frame | >11M Gaussians |
| Latent Gaussian Memory | ∼100 MB | – | 76.34 ms / frame | dual-level pruning |
| Dynamic Deviation Archive | 0.6 MB / frame | ∼22 GB | 0.266 ms / step | sampled CPU-to-GPU transfer |

Table 7: **Scaling to longer videos.** We report total inference time using our 14B model on one NVIDIA H200 GPU.

| Sequence length | Total time | Average time / frame |
|---|---|---|
| 81 frames | 14.0 min | 10.4 s |
| 301 frames | 55.3 min | 11.0 s |

**Scaling to longer videos.** The computational cost of *Robust Dreamer* scales linearly with the number of generated frames. Standard full-sequence video generation models often rely on bidirectional attention, where each frame attends to every other frame, leading to $O(T^2)$ temporal attention cost for a $T$-frame sequence. In contrast, our autoregressive Dreamer predicts one new frame at a time and conditions only on the anchor frame, the previous frame, the recalled memory context, and the current control, as formulated in Eq. 1. It therefore avoids attention over all generated frames and has $O(T)$ rollout complexity. Memory writing and recall are also performed once per generated frame, with the representation size controlled by the pruning strategy in Sec. A. Empirically, generating an 81-frame video takes about 14 minutes, while generating a 301-frame video takes 55.3 minutes on one NVIDIA H200 GPU, as shown in Tab. 7. This near-linear scaling indicates that the memory mechanism does not introduce disproportionate overhead for longer videos.

**Robustness to noisy geometry, occlusion, and dynamics.** The Dynamic Deviation Archive stores latent-space deviations synthesized by the current generator and reuses them to corrupt historical latents during training. These deviation-corrupted histories expose the Dreamer to imperfect memory states similar to those encountered at inference, improving robustness to drift, noisy geometry, occlusion, and dynamic disturbances. However, when reconstruction quality is severely degraded and recoverable geometry is no longer available, our method may still fail, similar to other 3D-aware world modeling approaches that fundamentally rely on usable geometry. We discuss this limitation further in Sec. G.

**Non-rigid dynamics.** Scenes with significant non-rigid motion are challenging because dynamic deviation patterns can be difficult to distinguish from true scene dynamics. Our method uses 4DGS-based temporal memory (Sec. B) to model dynamic content, and prior 4DGS studies suggest that Gaussian-based space-time representations can capture non-rigid motion in practice [44, 65, 71]. More specialized Gaussian-based approaches further demonstrate this potential. MD-Splatting [21] learns metric deformations for tracking highly deformable objects such as cloth, using physics-inspired regularization. GaussFluids [18] reconstructs temporally continuous Lagrangian fluid particles with Gaussian densities and a density-based incompressibility constraint. SurgicalGaussian [69] reconstructs deformable soft tissues from endoscopic videos using a forward-mapping deformation MLP and local motion regularization. These reconstruction results motivate deformation-aware and physics-guided extensions of our temporal memory; their effectiveness in action-controlled autoregressive generation remains to be established.

Nevertheless, our current 4DGS component does not fully solve extreme non-rigid dynamics. The primary focus of this paper is to address drift from latent–RGB cycling and training–inference mismatch, rather than to design a dedicated module for highly non-rigid phenomena such as fluids, fire, or topology-changing motion. Our current training and experiments mainly cover static scenes and dynamic but relatively structured motion. Stronger performance on highly non-rigid scenarios would likely require dedicated datasets with reliable pose annotations and explicit training on such cases, which remain scarce. We view systematic evaluation and modeling of such dynamics as important future work.

**Scope of the training–inference gap analysis.** We do not intend to attribute all long-horizon failures solely to the training–inference gap. Model capacity, scene complexity, control difficulty, reconstruction quality, and occlusion can also affect long-horizon generation. Our analysis in Sec. 3.3 focuses on accumulated drift as one important failure mode of autoregressive rollout. The fact that drift can still occur with a high-capacity 14B backbone suggests that scaling model capacity alone is insufficient. We emphasize the training–inference gap because it is a common issue in autoregressive video generation, as also discussed by Self-Forcing [28] and Diffusion Forcing [4]. Deviation Learning specifically addresses this factor by exposing the model to error-corrupted memory states during training, thereby reducing accumulated drift at inference time. Improving model capacity and handling more complex scenarios remain important and complementary directions.

Table 8: **Additional simple corruption baselines.** We replace our synthesized deviation with simple corruptions under the same setting.

| Method | PSNR ↑ | SSIM ↑ | LPIPS ↓ | FID ↓ |
|---|---|---|---|---|
| Blur | 14.13 | 0.271 | 0.391 | 47.80 |
| Color shift | 14.31 | 0.276 | 0.387 | 47.40 |
| Ours | **16.89** | **0.651** | **0.351** | **16.82** |

**Faithfulness of the one-step approximation.** We quantified the one-step deviation approximation against ODE-based simulation and found that the synthesized deviations reach about 58.7% of the error magnitude obtained by explicit ODE simulation. As discussed in Sec. 3.3, the most accurate way to obtain deviations is to simulate inference by solving the ODE. However, this is computationally prohibitive during training, because it requires running the full iterative sampling chain, e.g., 50 or 100 integration steps, for every historical frame in each training iteration. Faithfully matching multi-step diffusion rollout errors with only a one-step approximation is therefore inherently difficult.

This motivates the Dynamic Deviation Archive: rather than relying on a single approximation to perfectly reproduce full rollout error, we continuously collect diverse deviations throughout training. The archive uses an $\ell_2$-distance-based update rule to remove near-duplicate deviations while preserving diverse error modes. As a result, the model is exposed to a broad spectrum of realistic drift patterns and learns to recover from unexpected deviations more robustly at inference time. The ablation results in Tab. 3 (Row C vs. Row F) validate that this deviation learning design is effective in practice.

**Large deviations and hallucinated directions.** When the input is heavily corrupted, the one-step predicted velocity may point in an unreliable or hallucinated direction. This is one reason we maintain the Dynamic Deviation Archive instead of relying on a single deterministic approximation. The archive is intended to collect a diverse set of realistic error patterns arising during generation, including mild deviations as well as more severe or hallucination-like failures. The one-step approximation therefore does not need to be exact for every corrupted input; it needs to provide diverse corrupted conditions for training. The training objective in Eq. 4 then encourages the Dreamer to map deviation-corrupted conditions back to the clean ground-truth target, forcing the model to learn an internal correction mechanism. If the archive contained only simple or homogeneous deviations, the model would be exposed to a much narrower set of failure modes and would be more likely to drift under harder corrupted inputs at inference time. That said, if the corruption becomes extremely large or falls far outside the archive distribution, the current approximation may still fail. We regard this as a limitation and an important direction for future work.

**Simple corruption baselines.** Gaussian noise is a common corruption baseline, so our main ablation reports this variant in Tab. 3. Following this concern, we additionally test two simple corruption baselines, blur and color shift, by replacing our synthesized deviation with each corruption under the same setting. As shown in Tab. 8, these simple augmentations remain substantially worse than our method. This indicates that the archive does more than inject generic corruption: it stores structured, model-induced deviation patterns that better match autoregressive rollout errors.

# F More Related Work

**Feed-forward Reconstruction.** Foundational models like DUSt3R and MASt3R [60, 33] have achieved promising results by directly estimating point clouds from image pairs in a single pass, effectively handling challenging low-texture regions. To scale this capability to video sequences, Fast3R [70] extends processing to thousands of frames simultaneously, while VGGT [56] introduces a generalized framework capable of extracting multiple 3D attributes from variable input lengths. StreamVGGT [87] caches historical keys and values in a causal transformer framework to maintain a persistent representation over long horizons without exploding computational costs; however, its computational and memory usage still grow redundantly over time. Recurrent architectures like CUT3R [58] maintain a constant-sized memory state to ensure low inference costs, but they often suffer from forgetting earlier frames, leading to performance degradation as the sequence length increases. By implementing a simple yet efficient modification to the CUT3R architecture, TTT3R [9] explores a closed-form state update rule that enhances length generalization, allowing the model to reason over thousands of views while keeping memory and computation costs consistently low.

**Gaussian Splatting.** Gaussian Splatting has recently emerged as a powerful representation for real-time neural rendering. Kerbl et al.[30] pioneered 3D Gaussian Splatting (3DGS), which models scenes using collections of 3D Gaussians, achieving real-time high-resolution rendering with state-of-the-art visual quality. This breakthrough has sparked widespread interest and numerous extensions. Several works [65, 44, 71, 20, 19] have adapted 3DGS to dynamic scenes. Beyond novel-view synthesis, Gaussian-based and closely

related 3D field representations have also been explored for surface reconstruction [26, 79]: NeuSG [6] jointly optimizes Gaussian splatting and neural implicit surfaces, while VCR-GauS [8] couples depth and normal regularization with multi-view confidence. Semantic and open-vocabulary extensions [47, 5, 61] further enrich scene understanding, while ChatSplat [7] learns a 3D language field for object-, view-, and scene-level conversation. These Gaussian reconstruction methods typically require per-scene optimization. To improve cross-scene generalization, several studies [3, 10, 11, 84, 80, 53, 62] proposed feed-forward 3DGS frameworks that reconstruct scenes from sparse-view images by predicting pixel-aligned Gaussian parameters and unprojecting them into 3D space, supervised by the interpolated views. These approaches, however, depend on accurate camera poses and significant pose overlap. To mitigate this, NoPoseSplat [72] and Splatt3R [50] integrate Dust3r-based [60, 32] geometry estimation to eliminate pose dependency.

# G Limitation

Our method shares a common limitation with many 3D-aware world modeling approaches in that its performance depends on the quality of the underlying 3D reconstruction. When reliable geometry cannot be recovered—due to severe occlusions, reflective surfaces, rapid motion, or insufficient viewpoint diversity—the Latent Gaussian Memory may become inaccurate or incomplete, which in turn can degrade long-horizon generation quality. While our framework mitigates temporal drift through robust memory usage and deviation-aware training, it does not remove the fundamental dependency on successful 3D reconstruction. In addition, our framework is not fully end-to-end, as it separates memory construction and generative modeling into distinct components. This modular design improves interpretability and stability but may limit joint optimization across memory and generation. An interesting direction for future work is to explore more unified architectures, potentially leveraging diffusion models to jointly perform memory formation and long-horizon generation within a single end-to-end framework.

Our current 4DGS-based memory also remains limited under extreme non-rigid or topology-changing motion, such as fluids, fire, and highly deformable objects, because persistent Gaussian attributes may not capture their rapidly changing geometry and appearance. Time-varying illumination presents a related challenge: cast shadows are view- and illumination-dependent effects rather than persistent scene attributes, so storing and reprojecting them can produce inconsistent shadows and visible flicker across frames. Addressing these cases may require representations that more explicitly disentangle geometry, material appearance, dynamics, and lighting.

# H Impact Statements

This work focuses on algorithmic advances in long-horizon world modeling and does not involve the collection of new data. All training and evaluation are conducted using publicly available datasets, without the use of any private, sensitive, or personally identifiable data. The proposed methods are primarily designed for scene-level and environment-centric modeling, rather than human-centered applications. As a result, we do not anticipate direct ethical or societal risks related to privacy, data ownership, or human subject impact. As with other generative video and world-modeling techniques, potential misuse could include generating misleading scene-level simulations or over-relying on simulated rollouts in downstream decision-making, so deployment should be paired with dataset provenance checks and application-specific validation. We believe the contributions of this work are largely technical and methodological, supporting future research in simulation and world modeling under responsible research practices.

# I More Qualitative Results

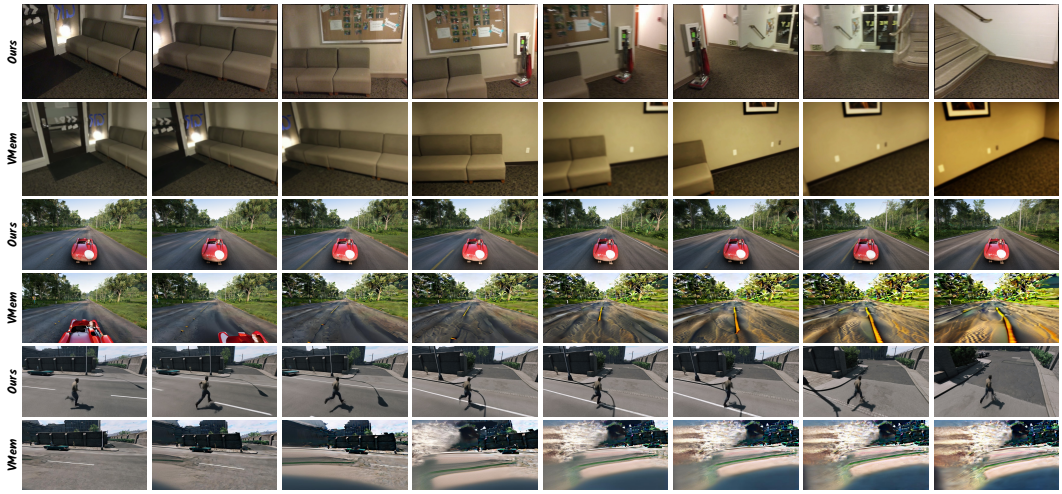


Figure 6: **Qualitative results on static long scenes from ScanNet (300 frames) and dynamic scenes from OmniWorldGame (80 frames).** The top two rows show experiments on ScanNet, while the bottom four rows present comparisons on OmniWorldGame. Displayed frames are randomly sampled from the early, middle, and late stages of the sequences. Compared to the state-of-the-art baseline VMem, which also utilizes a 3D memory mechanism, our approach successfully avoids the color drift issue. This demonstrates the effectiveness of our proposed latent-memory inheritance and Deviation Learning.



Figure 7: **Qualitative results on OmniWorldGame.** We visualize 64 randomly sampled frames from an 80-frame **dynamic scene**.

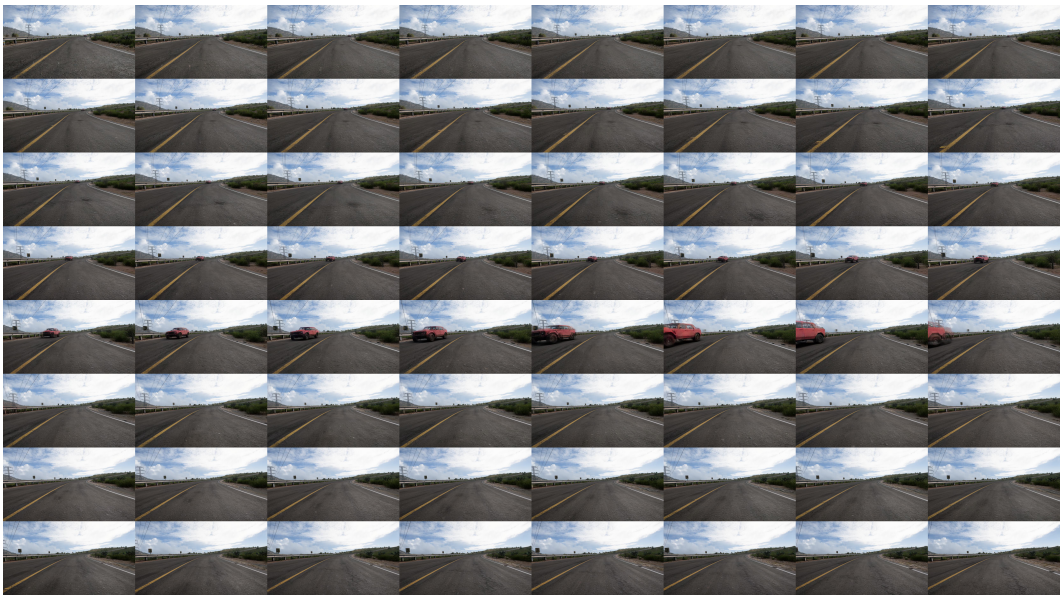

Figure 8: **Qualitative results on OmniWorldGame.** We visualize 64 randomly sampled frames from an 80-frame **dynamic scene**.



Figure 9: **Qualitative results on OmniWorldGame.** We visualize 64 randomly sampled frames from an 80-frame **dynamic scene**.

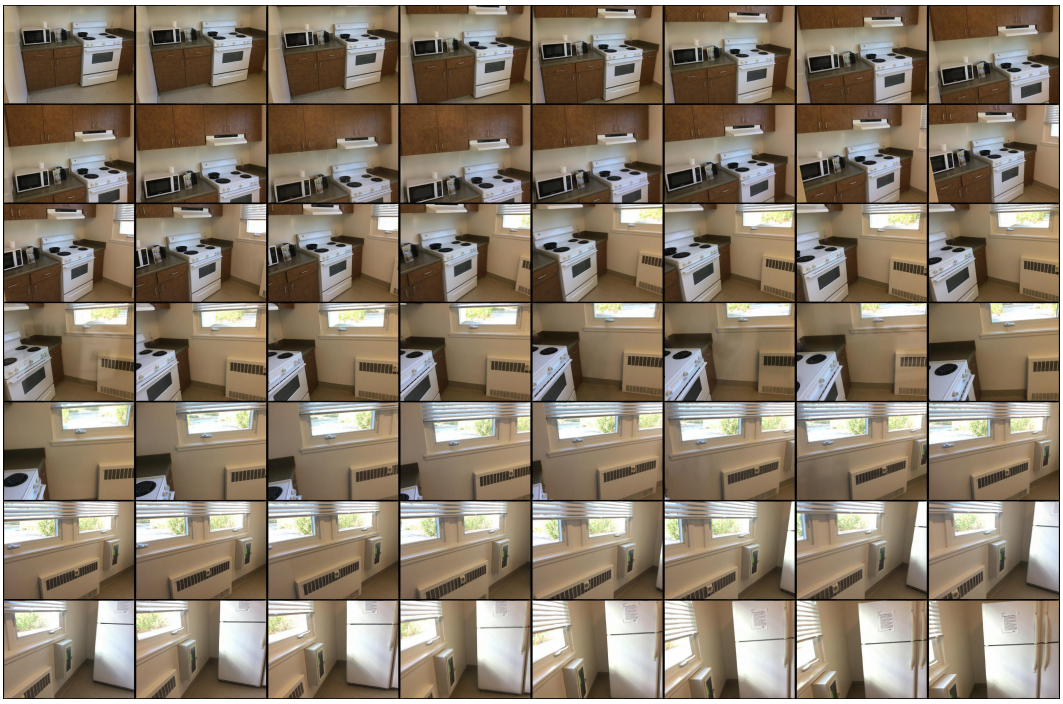

Figure 10: **Qualitative results on ScanNet.** We visualize 56 frames uniformly sampled from a 300-frame sequence of a **long static scene**.

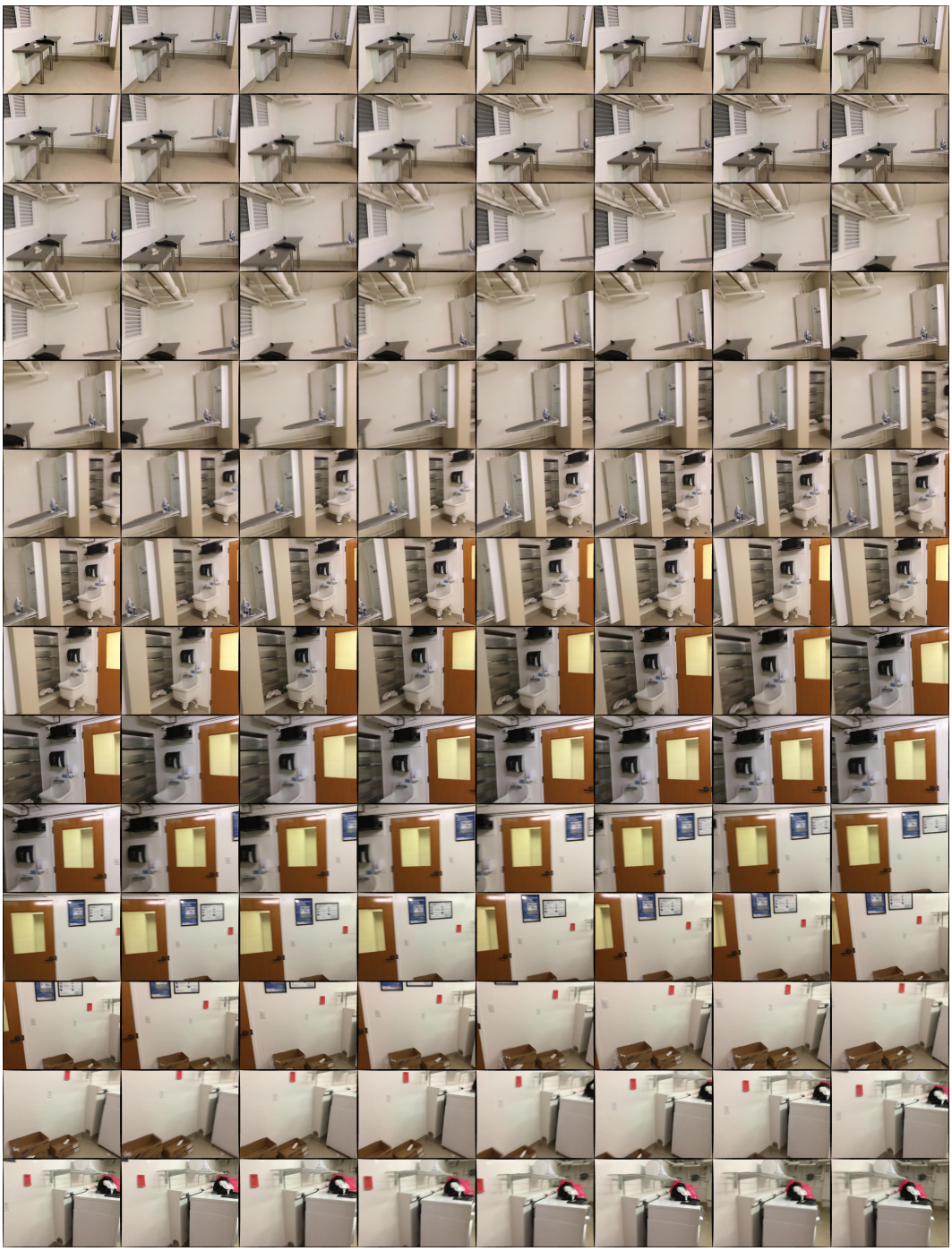

Figure 11: **Qualitative results on ScanNet.** We visualize 112 frames uniformly sampled from a 300-frame sequence of a **long static scene**.

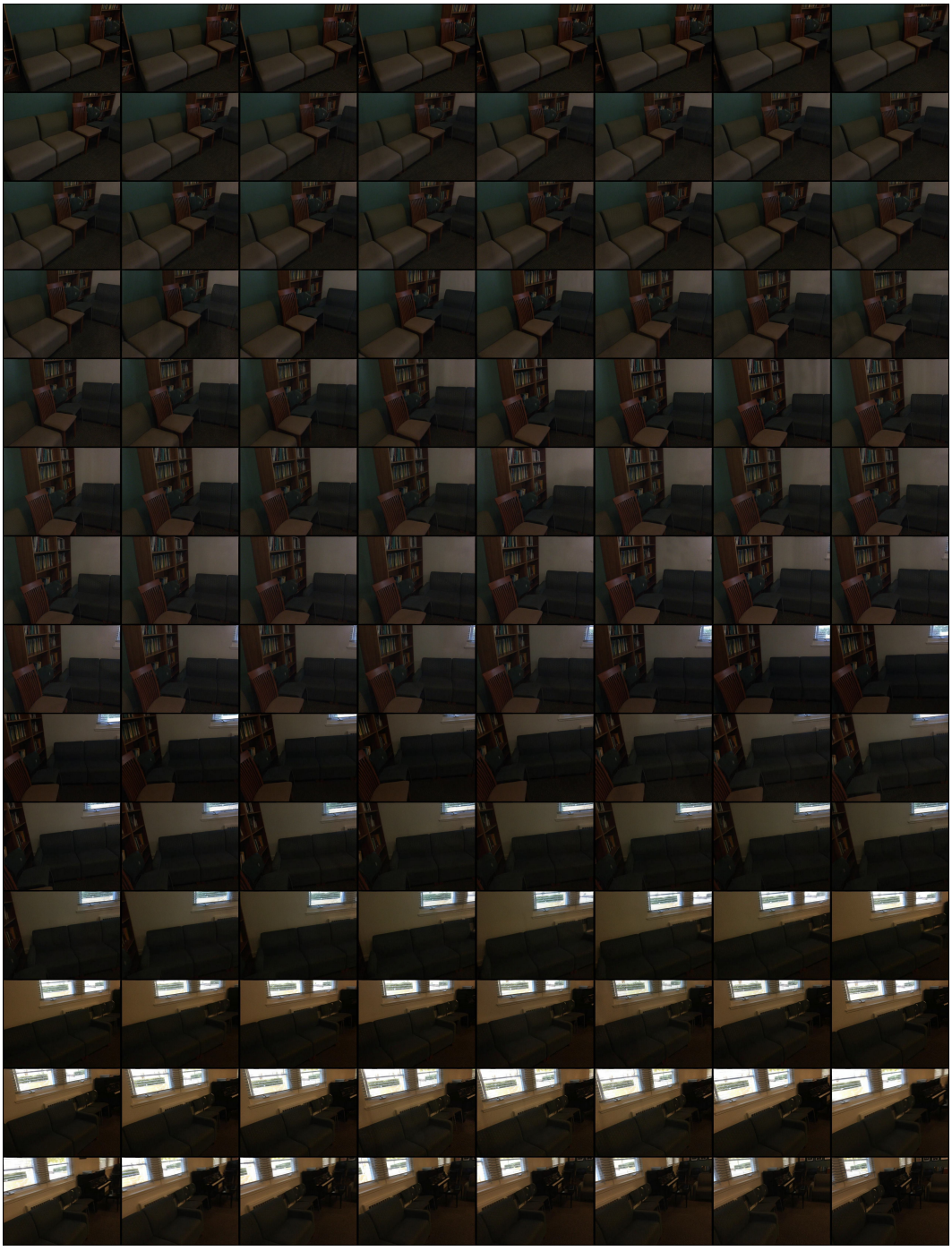

Figure 12: **Qualitative results on ScanNet.** We visualize 112 frames uniformly sampled from a 300-frame sequence of a **long static scene**.

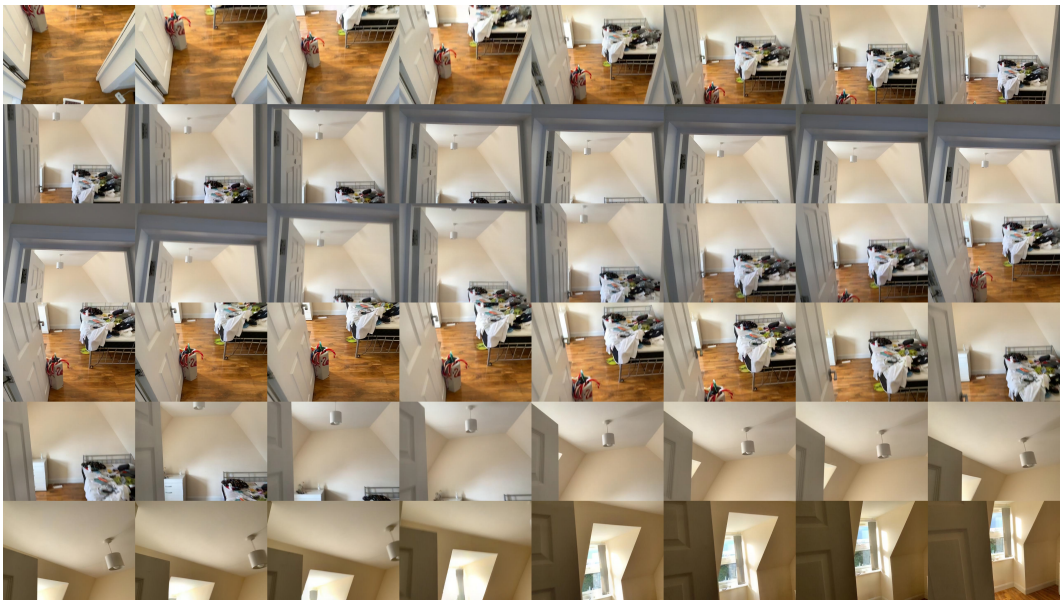

Figure 13: **Qualitative results on an out-of-domain scene.** We visualize 48 uniformly sampled frames from an 80-frame sequence to demonstrate generalization.



Figure 14: **Qualitative results on an out-of-domain scene.** We visualize 40 uniformly sampled frames from an 80-frame sequence to demonstrate generalization.



Figure 15: **Qualitative results on an out-of-domain dynamic scene.** We visualize 64 uniformly sampled frames from an 80-frame sequence to demonstrate generalization.

# NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Sec. 1 state the scope, assumptions, and contributions of Robust Dreamer. The method in Sec. 3 and experiments in Sec. 4 support the claimed improvements in long-horizon action-controlled video generation.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The appendix includes a dedicated Limitation section discussing dependence on 3D reconstruction quality and the modular, not fully end-to-end, design. These limitations clarify failure cases such as occlusion, reflective surfaces, rapid motion, and insufficient viewpoint diversity.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results or proofs. The equations in Sec. 3 define the model, memory, archive, and training objective rather than theorem statements.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper specifies the method, training objective, datasets, metrics, baselines, implementation details, hyperparameters, and compute resources in Sec. 3, Sec. 4.1, Sec. C, and Tab. 4. These details provide the information needed to reproduce the main experimental comparisons at the level described in the submission.

   Guidelines:

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

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

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

   Answer: [No]

   Justification: The experiments use publicly available datasets, but the current submission does not include an open code release, anonymized repository, or command-level reproduction instructions. We plan to provide release instructions in a future version where possible.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Sec. 4.1 describes datasets, metrics, evaluation settings, and baselines, while Sec. C and Tab. 4 provide model, optimization, training, resolution, sequence-length, and deviation-learning details.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports aggregate quantitative metrics in Tab. 1, Tab. 2, and Tab. 3, but does not include error bars, confidence intervals, or statistical significance tests. Repeated runs are computationally expensive for the 14B-scale video model, so this version focuses on fixed-protocol comparisons.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Sec. C reports the use of 8 NVIDIA H200 GPUs with 144 GB memory each, the approximate 31-hour runtime for each training stage, and the inference time for generating an 81-frame video.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available datasets and does not collect private, sensitive, or personally identifiable data, as discussed in the Impact Statements. The work is methodological and environment-centric rather than a human-subject study.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The Impact Statements discuss the positive role of the method in responsible world-modeling research and explain why direct privacy, data ownership, and human-subject risks are limited. They also discuss possible misuse through misleading scene-level simulations or over-reliance on simulated rollouts in downstream decision-making.

    Guidelines:

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

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not introduce a scraped dataset or release a high-risk pretrained generative model in the current submission. The method is evaluated on existing public datasets, and no new high-risk asset release is described.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: The paper cites the datasets, model backbones, and software components used in the work, including ScanNet, DL3DV, OmniWorldGame, Wan2.1, CUT3R-style reconstruction, and gsplat. The current manuscript does not yet explicitly list the licenses and terms of use for each asset.

    Guidelines:

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset or other standalone research asset in the current submission. It proposes a method and evaluates it using existing datasets and model components.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. It uses existing public datasets for training and evaluation.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval or equivalent review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method does not use LLMs as an important, original, or non-standard component. The paper builds on video diffusion, latent Gaussian memory, and deviation-aware training rather than LLM-based methodology.

    Guidelines:

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