DEMIURGE: FORGING NAVIGABLE 4D WORLDS VIA AUTOREGRESSIVE VIDEO RERENDERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in video diffusion models have substantially empowered generative video rerendering, where novel-view videos can be synthesized from a single input video, facilitating a broad spectrum of applications such as world modeling, embodied intelligence, and robotic simulation. However, despite their impressive photorealism, existing video rerendering methods typically treat each camera trajectory in isolation, lacking the cross-trajectory consistency required to form a coherent and traversable 4D world. To address this limitation, we introduce ***Demiurge***, a novel autoregressive video rerendering pipeline for navigable 4D world generation. Built upon a dynamic 4D point-cloud memory cache, Demiurge promotes cross-trajectory consistency through **four key components: 1)** an appearance video diffusion model that rerenders photorealistic novel-view videos along target camera trajectories conditioned on the 4D memory; **2)** a memory-consistent video depth completion model that aligns known-region depths with cached geometry while completing missing regions, mitigating scale ambiguity and discontinuities during memory updates; **3)** a memory-aware dual-stream warping strategy that provides high-fidelity visual guidance and occlusion cues for plausible novel-view video synthesis; **4)** a geometry-aware video refinement model that further enhances the perceptual quality of synthesized videos. Extensive experiments show that Demiurge achieves state-of-the-art performance with **three key capabilities: 1)** coherent spatiotemporal lattice construction over dense multi-view trajectories; **2)** large-parallax video rerendering under challenging camera motions, such as 180° rotations with no input-view overlap; **3)** geometry-faithful outputs that can be directly optimized into high-quality 3D Gaussian Splatting representations with minimal artifacts. The anonymous project page is available at `https://anonymous.4open.science/w/Demiurge/`.

1 INTRODUCTION

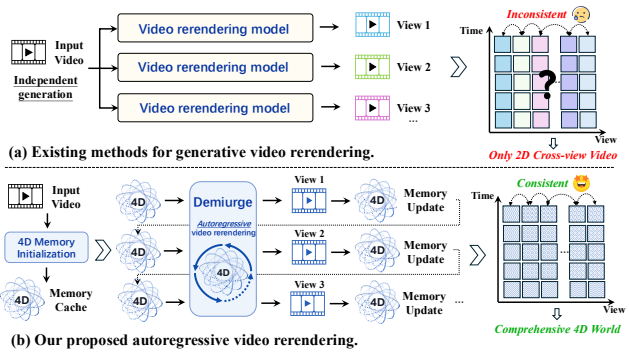


Figure 1: **Comparison of our proposed method with existing methods. (a)** Existing methods independently synthesize videos along different target trajectories, resulting in cross-trajectory inconsistencies and only isolated 2D cross-view videos. **(b)** In contrast, Demiurge autoregressively generates each trajectory with a shared 4D memory, maintaining cross-trajectory consistency and thereby enabling coherent 4D world construction.

As a fundamental medium for representing evolving visual content, videos provide a natural record of real-world dynamics, serving as a cornerstone for information sharing, content creation, and scene

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

understanding Tong et al. (2022); Ho et al. (2022). However, despite such informative spatiotemporal cues, a single video alone still falls short of fully representing the underlying physical world, as it provides only 2D projections captured along a specific camera trajectory, preserving partial observations of the scene structure while leaving occluded regions unobserved. This inherent limitation has spurred growing interest in generative video rerendering, which aims to synthesize novel-view videos from a single input video along target camera trajectories, offering a promising pathway towards world modeling Yu et al. (2024; 2025b), embodied intelligence Tu et al. (2026), and robotic simulation Zhou et al. (2024).

To achieve this goal, leveraging recent advances in video diffusion models, existing methods typically adopt a camera-conditioned video generation paradigm, incorporating target camera trajectories through intrinsics/extrinsics embeddings Wang et al. (2024), Plücker ray maps Bahmani et al. (2025b), or depth-based warping cues Lin et al. (2026); Zhu et al. (2026). However, as shown in Fig. 1(a), while photorealistic novel-view videos can be obtained along individual camera paths, the output for each trajectory is synthesized independently, leaving scene geometry, appearance, and dynamics across different views unconstrained. Consequently, the synthesized observations cannot be consolidated into a globally consistent spatiotemporal lattice, precluding the formation of a coherent and traversable 4D world.

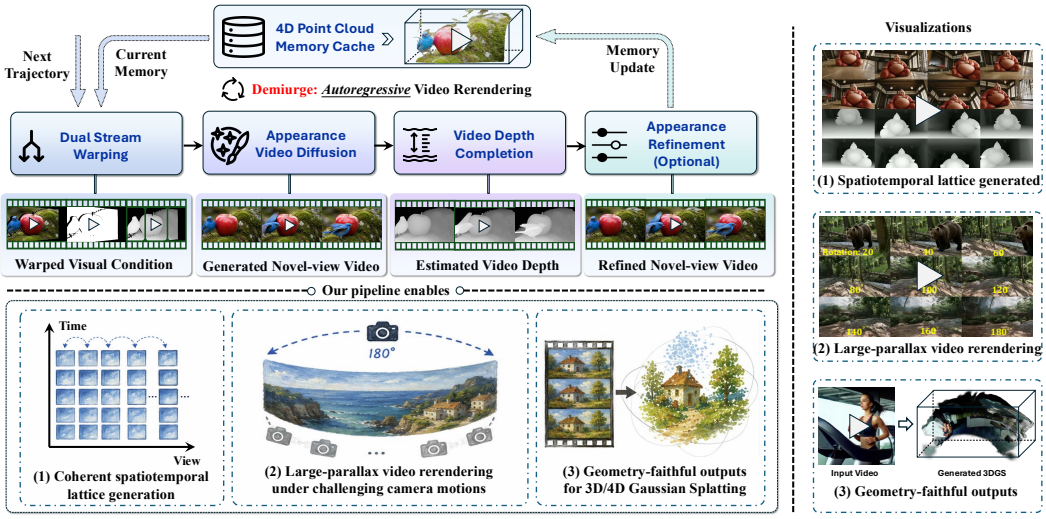


Figure 2: **Overview of Demiurge.** Demiurge performs autoregressive video rerendering with an evolving 4D point-cloud memory through a **generate–recover–update cycle**. By integrating dual-stream warping, appearance generation, depth completion, and optional refinement, it enables coherent 4D world construction, large-parallax rerendering, and geometry-faithful 3D/4D reconstruction.

To move beyond trajectory-wise synthesis towards navigable 4D world construction, as shown in Fig. 1(b), we introduce ***Demiurge***, an autoregressive video rerendering framework underpinned by a dynamic 4D point-cloud memory. At each autoregressive step, a novel-view video is synthesized along a target camera trajectory using the current memory as conditioning, with its appearance and recovered geometry subsequently fused back into the 4D memory to guide subsequent rerendering and thus maintain cross-trajectory consistency. Building upon this formulation, as shown in Fig. 2, **Demiurge comprises four key components: 1) a memory-guided appearance video diffusion model** that synthesizes photorealistic novel-view videos along target camera trajectories using point-cloud renderings from the 4D memory, complemented by a lightweight noise-monitoring mechanism that detects and suppresses conspicuous outliers in the rendered guidance to provide cleaner conditioning signals; **2) a memory-consistent video depth completion model** that recovers temporally coherent dense depth maps for each synthesized novel-view video under the guidance of partial depth renderings from the 4D memory, anchoring predictions in previously observed regions to cached metric geometry while completing newly revealed regions, thereby allowing the recovered depths to be back-projected into dynamic point clouds and reliably fused into the memory for geometrically consistent memory updates; **3) a memory-aware dual-stream warping strategy** that derives high-fidelity appearance guidance from a selectively updated point cloud and reliable occlu-

sion cues from a densely updated counterpart, fusing the two according to their depth discrepancies to preserve stable visual content and guide plausible synthesis in newly exposed regions, thereby suppressing geometric artifacts and cumulative color drift over autoregressive iterations; **4) an optional geometry-aware video refinement model** that leverages temporally consistent target-view depth maps alongside partial RGB renderings to attenuate subtle texture flicker and noise, yielding refined outputs with enhanced visual fidelity and temporal coherence.

Extensive evaluations spanning diverse scenes and viewing paths demonstrate that Demiurge attains state-of-the-art results in perceptual realism, geometric accuracy, and cross-trajectory consistency. Notably, it delivers **three capabilities** essential to navigable 4D worlds: **1)** assembling coherent spatiotemporal lattices over densely sampled viewpoints; **2)** sustaining robust rerendering under extreme parallax, including 180° rotations without input-view overlap; and **3)** producing geometry-faithful videos that can be directly optimized into high-quality 3D Gaussian Splatting representations with minimal artifacts.

Our main contributions are summarized as follows:

- We propose ***Demiurge***, an autoregressive video rerendering framework with a dynamic 4D point-cloud memory for cross-trajectory-consistent and navigable 4D world generation.
- To ensure high-fidelity synthesis and consistent memory evolution, **we introduce four coordinated designs:** an appearance video diffusion model for photorealistic novel-view synthesis; a video depth completion model for reliable geometry recovery and memory updates; a dual-stream warping strategy for appearance-preserving occlusion handling without cumulative color drift; and an optional video refinement model for attenuating texture flicker and noise.
- Extensive experiments demonstrate that Demiurge achieves state-of-the-art perceptual realism and cross-trajectory consistency, supports robust large-parallax rerendering, and yields geometry-faithful outputs for high-quality 3D Gaussian Splatting reconstruction.

2 Related Works

2.1 Camera-Controllable Video Generation

Early methods condition video diffusion on parameterized camera poses, either through a dedicated camera-motion module Wang et al. (2024) or via pixel-aligned Plücker ray embeddings He et al. (2024); Xu et al. (2024); Bahmani et al. (2025a;b); He et al. (2025); Zheng et al. (2024). However, since viewpoint transformations are learned implicitly, they are prone to geometric distortion and content drift under large camera motion. Recently, geometry-grounded methods Gu et al. (2025); Ma et al. (2025) instead lift source observations into 3D and reproject them into target views. For example, ViewCrafter Yu et al. (2024) and GEN3C Ren et al. (2025) condition on partial point-cloud renderings, FlexWorld Chen et al. (2025b) and Voyager Huang et al. (2025) progressively expand persistent 3D scenes with newly generated observations, and GTA Zhu et al. (2026) decouples geometry generation from appearance synthesis. More recently, camera control has been repurposed as a generative renderer for dynamic scenes Zheng et al. (2026); Yang et al. (2026); Song et al. (2026); Lin et al. (2026); Park et al. (2026), re-rendering monocular videos along novel trajectories while preserving their original dynamics, either from encoded extrinsics Bai et al. (2025) or by explicitly warping the source video with estimated geometry Yu et al. (2025b). Both, however, synthesize each trajectory independently and retain nothing for subsequent views. In contrast, Demiurge maintains an explicit 4D memory of generated appearance and recovered geometry, enforcing long-range consistency across diverse camera trajectories.

2.2 Memory-Augmented Video Generation

To mitigate context forgetting and visual drift in long-horizon generation, one line of work retrieves or reuses historical frames according to their relevance to the current state Xiao et al. (2026); Yu et al. (2025a); Schneider et al. (2025); Chen et al. (2025a), while another compresses the generation history into hierarchical, multi-scale, or sliding-window latent memories Liu et al. (2026); Li et al. (2025a); Mao et al. (2025); Cai et al. (2026); Team et al. (2026). Such appearance- and feature-space memories offer limited explicit geometric grounding, causing spatial drift under large camera motion. Geometry-aware approaches instead organize history in explicit 3D space, using a surfel-indexed view memory Li et al. (2025b), point-cloud spatial context Wu et al. (2026), an incrementally expanded RGB-D cache Huang et al. (2025), or a persistent point cloud with dynamic–static separation Zhao et al. (2026); yet they predominantly preserve static geometry rather than the

complete time-varying scene state. Most closely related to ours, PlenopticDreamer Fu et al. (2026) banks previously generated video–camera pairs and retrieves the top-k by 3D field-of-view overlap. Its fixed context capacity, however, forces selective retrieval or trajectory fusion once the number of trajectories exceeds k, discarding fine-grained detail and tying long-range consistency to retrieval quality. In contrast, Demiurge consolidates reliable appearance and recovered geometry from all generated trajectories into a unified 4D memory, avoiding a fixed video-context bottleneck.

3 METHOD

Overview. Given an input video $\mathbf{V}^0 = \{\mathbf{I}_t^0\}_{t=1}^T$ and its camera trajectory $\mathbf{C}^0 = \{\mathbf{C}_t^0\}_{t=1}^T$, our goal is to rerender the observed scene along N target trajectories $\{\mathbf{C}^i\}_{i=1}^N$, where t indexes the T video frames and i indexes the target trajectories. To this end, Demiurge maintains an evolving 4D point-cloud memory $\mathcal{M}_i = \{\mathcal{P}_{i,t}\}_{t=1}^T$ as a persistent world state that retains historical geometry and appearance across autoregressive iterations, where $\mathcal{P}_{i,t}$ denotes the colored 3D point cloud accumulated at time t after processing the i-th trajectory. The initial memory $\mathcal{M}_0$ is reconstructed from the input video using a monocular video depth estimator. As illustrated in Fig. 2, at iteration i, the current memory $\mathcal{M}_{i-1}$ is rendered (via dual-stream warping in Sec. 3.3) along $\mathbf{C}^i$ to guide the synthesis of a novel-view video $\mathbf{V}^i = \{\mathbf{I}_t^i\}_{t=1}^T$ (via appearance video diffusion in Sec. 3.1) and its temporally coherent dense depth maps $\mathbf{D}^i = \{\mathbf{D}_t^i\}_{t=1}^T$ (via video depth completion in Sec. 3.2). The generated appearance and recovered geometry are subsequently back-projected and fused into the memory to obtain $\mathcal{M}_i$, which guides the next trajectory. By iterating this generate–recover–update process, Demiurge progressively constructs a navigable 4D world. Further details on the 4D memory representation and point-cloud warping are provided in Appendix A.

3.1 MEMORY-GUIDED APPEARANCE VIDEO DIFFUSION

Constructing a navigable 4D world requires progressively expanding the observed scene along new camera trajectories while preserving the appearance and geometry accumulated from previous observations. This, in turn, calls for a generative model capable of synthesizing photorealistic videos along a target trajectory while remaining grounded in the current world state. To this end, at the i-th autoregressive step, we first render the 4D memory $\mathcal{M}_{i-1}$ along the target trajectory $\mathbf{C}^i$ to obtain a sequence of memory-derived visual conditions:

$$\left(\widetilde{\mathbf{V}}^i, \widetilde{\mathbf{D}}^i, \widetilde{\mathbf{M}}^i\right) = \mathcal{R}_{\text{dual}}\left(\mathcal{M}_{i-1}, \mathbf{C}^i\right), \tag{1}$$

where $\mathcal{R}_{\text{dual}}$ denotes our dual-stream warping operator presented in Sec. 3.3 (Eq. 13), and $\widetilde{\mathbf{V}}^i$, $\widetilde{\mathbf{D}}^i$, and $\widetilde{\mathbf{M}}^i$ respectively denote the resulting partial RGB video, partial depth video, and visibility masks, providing complementary appearance, geometry, and visibility cues.

However, discrete point-cloud rendering may inherently produce holes and unstable projections (as shown in Fig. 4(a)), particularly near occlusion boundaries, which can accumulate over successive autoregressive iterations. To prevent such corrupted observations from conditioning the subsequent generation process, we introduce a lightweight, non-learned noise monitor that identifies potentially unreliable pixels from the rendered depth. Specifically, for each pixel $\mathbf{p}$ in the depth frame $\widetilde{\mathbf{D}}_t^i$, we define

$$\mathbf{O}_t^i(\mathbf{p}) = \mathbb{I}\Big[\widetilde{\mathbf{D}}_t^i(\mathbf{p}) < \tau_0 \ \vee \ \Big\|\nabla\widetilde{\mathbf{D}}_t^i(\mathbf{p})\Big\|_2 > \tau_g \vee \ \mathrm{Var}_{\mathbf{q}\in\mathcal{N}_3(\mathbf{p})}\left(\widetilde{\mathbf{D}}_t^i(\mathbf{q})\right) > \tau_v\Big], \tag{2}$$

where $\mathcal{N}_3(\mathbf{p})$ denotes the 3×3 neighborhood centered at $\mathbf{p}$, and τ_0, τ_g, and τ_v are thresholds for invalid depth, depth-gradient magnitude, and local depth variance, respectively. The resulting sequence $\mathbf{O}^i = \{\mathbf{O}_t^i\}_{t=1}^T$ captures missing depth values, abrupt depth variations, and locally unstable projections. We consistently invalidate these pixels across all rendered conditions:

$$\left(\overline{\mathbf{V}}^i, \overline{\mathbf{D}}^i, \overline{\mathbf{M}}^i\right) = \left(1 - \mathbf{O}^i\right) \odot \left(\widetilde{\mathbf{V}}^i, \widetilde{\mathbf{D}}^i, \widetilde{\mathbf{M}}^i\right). \tag{3}$$

Given the cleaned conditions, our memory-guided appearance video diffusion model $\mathcal{F}_{\text{app}}$ synthesizes the target novel-view video as

$$\mathbf{V}^i = \mathcal{F}_{\text{app}}\left(\overline{\mathbf{V}}^i, \overline{\mathbf{D}}^i, \overline{\mathbf{M}}^i\right), \tag{4}$$

where $\mathbf{V}^i = \{\mathbf{I}_t^i\}_{t=1}^T$ denotes the generated RGB video along $\mathbf{C}^i$. Internally, the three conditions are concatenated with the noisy video latent during denoising. In this way, $\mathcal{F}_{\text{app}}$ preserves reliable memory-supported content while regenerating unreliable and newly exposed regions with photorealistic details. Further details are provided in Appendix B.1 and B.2.

3.2 MEMORY-CONSISTENT VIDEO DEPTH COMPLETION

Given the generated novel-view RGB video $\mathbf{V}^i$, to incorporate its newly observed content into the 4D memory and support the next autoregressive step, we need to recover a dense video depth that is both temporally coherent and geometrically consistent with the existing memory. A straightforward solution is to apply an off-the-shelf monocular video depth estimator to $\mathbf{V}^i$. However, monocular depth is inherently scale-ambiguous and may disagree with the cached geometry even in previously observed regions. Directly back-projecting such predictions can introduce misaligned surfaces and geometric discontinuities into the memory (as shown in Fig. 4(b)), which may further accumulate over subsequent iterations. To address this issue, we therefore recover dense geometry from the generated video through a memory-consistent, diffusion-based video depth completion model.

Specifically, our model takes the generated video $\mathbf{V}^i$ and the cleaned partial cached depth $\overline{\mathbf{D}}^i$ as inputs. The partial cached depth anchors the prediction to the accumulated geometry in previously observed regions, while the generated RGB video provides spatial and temporal cues for estimating the geometry of newly revealed content. Since the model is built upon latent video diffusion, we first convert the partial depth into disparity and normalize its valid values to $[0, 1]$:

$$\mathbf{U}^i = 1/(\overline{\mathbf{D}}^i + \epsilon),\ \overline{\mathbf{U}}^i = \mathcal{N}\left(\mathbf{U}^i; \overline{\mathbf{M}}^i\right), \tag{5}$$

where ϵ ensures numerical stability, and $\mathcal{N}(\cdot)$ denotes min–max normalization over valid pixels.

Conditioned on the generated RGB video and normalized partial disparity, our memory-consistent video depth completion model $\mathcal{F}_{\text{dep}}$ predicts a temporally coherent dense relative disparity video:

$$\widehat{\mathbf{U}}^i = \mathcal{F}_{\text{dep}}\left(\mathbf{V}^i, \overline{\mathbf{U}}^i\right). \tag{6}$$

To preserve memory-derived geometry throughout denoising, we employ a dual-path conditioning design within $\mathcal{F}_{\text{dep}}$. The partial cached disparity anchors the denoising process to previously observed geometry, while the generated RGB video provides complementary spatial and temporal cues for completing newly revealed regions. Detailed architecture is provided in the supplementary material.

While $\widehat{\mathbf{U}}^i$ captures the relative geometric structure of the target video, it does not preserve the metric scale of the memory. We therefore employ a two-stage global affine alignment. Specifically, we first align the prediction to the normalized partial cached depth and then recover its metric scale using the original cached depth. Both transformations are estimated over all memory-supported pixels and shared across frames, avoiding temporal scale fluctuations. We denote the resulting dense metric-depth prediction by $\mathbf{G}^i$.

Despite the global alignment, small local deviations may still remain in previously observed regions, while directly replacing them with cached depths can introduce boundary discontinuities. We therefore perform Poisson refinement Pérez et al. (2023) in metric-depth space:

$$\mathbf{D}^i = \arg\min_{\mathbf{X}^i} \quad \sum_{t=1}^{T} \left\|\nabla \mathbf{X}_t^i - \nabla \mathbf{G}_t^i\right\|_2^2, \text{s.t.} \quad \mathbf{X}_t^i(\mathbf{p}) = \overline{\mathbf{D}}_t^i(\mathbf{p}), \quad \forall\, \mathbf{p} \text{ with } \overline{\mathbf{M}}_t^i(\mathbf{p}) = 1. \tag{7}$$

This refinement preserves the spatial gradients of the aligned dense prediction while enforcing exact agreement with the cached geometry. The resulting dense depth video $\mathbf{D}^i$ is subsequently fused into the 4D memory together with $\mathbf{V}^i$. Further alignment and solver details are provided in the supplementary material. We defer further details to Appendix B.3.

3.3 MEMORY-AWARE DUAL-STREAM WARPING

After recovering the RGB-D video $(\mathbf{V}^i, \mathbf{D}^i)$, we next consider how to update the point-cloud memory so that it can provide reliable conditions for the subsequent target trajectory. A natural strategy is to register only newly generated pixels while retaining the points accumulated in previously observed regions. Such a selective update prevents stable colors from being repeatedly overwritten by diffusion-generated appearance, thereby preserving sharp and visually consistent renderings over successive autoregressive iterations.

However, selective updates may leave the point cloud insufficiently dense for reliable visibility reasoning (as shown in the leftmost RGB–depth pair of Fig. 4(d)). Under large viewpoint changes,

sparse foreground points may fail to fully occlude the background during z-buffer rendering, causing background content that should remain hidden to leak into foreground regions. This failure is not captured by the visibility mask, which only records whether a pixel originates from cached or newly generated content, rather than whether its projection remains geometrically valid under the target viewpoint. Consequently, these erroneous projections may be treated as reliable appearance guidance, propagated into the diffusion output, and eventually written back into the memory.

An alternative is to register all pixels from each generated RGB-D video. The resulting dense point cloud provides more complete foreground geometry and enables more reliable occlusion reasoning under large parallax. Nevertheless, this strategy also repeatedly incorporates the generated appearance of previously observed regions. Small texture deviations and high-frequency artifacts can therefore accumulate over multiple iterations, eventually producing blurred renderings and color drift (as shown in the middle RGB–depth pair of Fig. 4(d)).

To reconcile the competing requirements of appearance preservation and geometric completeness, we introduce **dual-stream warping**. Conceptually, the global 4D memory comprises two complementary memory states:

$$\mathcal{M}_i = (\mathcal{M}_i^{\text{app}}, \mathcal{M}_i^{\text{geo}}), \tag{8}$$

where the appearance memory $\mathcal{M}_i^{\text{app}}$ selectively incorporates newly synthesized regions to preserve stable colors, while the geometry memory $\mathcal{M}_i^{\text{geo}}$ densely incorporates all valid RGB-D observations to support reliable occlusion reasoning. Both memories are initialized from the same input video and subsequently evolve under their respective update policies:

$$\begin{aligned} \mathcal{M}_i^{\text{app}} &= \text{Fuse}\left(\mathcal{M}_{i-1}^{\text{app}}, \text{BackProj}\left(\mathbf{V}^i, \mathbf{D}^i, \mathbf{C}^i; 1 - \overline{\mathbf{M}}^i\right)\right), \\ \mathcal{M}_i^{\text{geo}} &= \text{Fuse}\left(\mathcal{M}_{i-1}^{\text{geo}}, \text{BackProj}\left(\mathbf{V}^i, \mathbf{D}^i, \mathbf{C}^i; \mathbf{1}\right)\right). \end{aligned} \tag{9}$$

Here, $\text{BackProj}(\cdot)$ lifts the selected RGB-D pixels into world-space colored point clouds using the corresponding camera parameters, while $\text{Fuse}(\cdot)$ appends the recovered points to the accumulated memory. The last argument of $\text{BackProj}(\cdot)$ denotes the pixel-selection mask.

For the next target trajectory $\mathbf{C}^{i+1}$, we render the two memory states in parallel, producing an appearance warping stream and a geometry warping stream:

$$\left(\mathbf{V}_{\text{app}}^{i+1}, \mathbf{D}_{\text{app}}^{i+1}, \mathbf{M}_{\text{app}}^{i+1}\right) = \mathcal{R}\left(\mathcal{M}_i^{\text{app}}, \mathbf{C}^{i+1}\right), \left(\mathbf{V}_{\text{geo}}^{i+1}, \mathbf{D}_{\text{geo}}^{i+1}, \mathbf{M}_{\text{geo}}^{i+1}\right) = \mathcal{R}\left(\mathcal{M}_i^{\text{geo}}, \mathbf{C}^{i+1}\right), \tag{10}$$

where $\mathcal{R}$ denotes point cloud rendering.

The appearance stream provides sharp and stable visual guidance, whereas the geometry stream offers denser geometric coverage for more reliable occlusion reasoning. We identify viewpoint-dependent projection conflicts by comparing their rendered depths:

$$\boldsymbol{\Delta}_t^{i+1}(\mathbf{p}) = \left|\mathbf{D}_{\text{app},t}^{i+1}(\mathbf{p}) - \mathbf{D}_{\text{geo},t}^{i+1}(\mathbf{p})\right|, \ \mathbf{Q}_t^{i+1}(\mathbf{p}) = \mathbb{I}\left[\boldsymbol{\Delta}_t^{i+1}(\mathbf{p}) > \tau_{\text{occ}}\right], \tag{11}$$

where $\mathbf{p}$ denotes a pixel location in the target image plane, $\boldsymbol{\Delta}_t^{i+1}(\mathbf{p})$ measures the depth discrepancy between the two streams at frame t, $\mathbf{Q}^{i+1}$ is the resulting conflict mask, and τ_{occ} is the discrepancy threshold. In regions without detected conflicts, we retain the RGB-D conditions from the appearance stream. At locations indicated by $\mathbf{Q}^{i+1}$, we instead adopt the geometry-stream conditions, whose denser point distribution provides reliable visibility reasoning:

$$\begin{aligned} \widetilde{\mathbf{V}}^{i+1} &= \left(1 - \mathbf{Q}^{i+1}\right) \odot \mathbf{V}_{\text{app}}^{i+1} + \mathbf{Q}^{i+1} \odot \mathbf{V}_{\text{geo}}^{i+1}, \\ \widetilde{\mathbf{D}}^{i+1} &= \left(1 - \mathbf{Q}^{i+1}\right) \odot \mathbf{D}_{\text{app}}^{i+1} + \mathbf{Q}^{i+1} \odot \mathbf{D}_{\text{geo}}^{i+1}, \\ \widetilde{\mathbf{M}}^{i+1} &= \left(1 - \mathbf{Q}^{i+1}\right) \odot \mathbf{M}_{\text{app}}^{i+1} + \mathbf{Q}^{i+1} \odot \mathbf{M}_{\text{geo}}^{i+1}. \end{aligned} \tag{12}$$

Equivalently, denoting the pixel-wise fusion defined in Eqs. 11 and 12 by $\mathcal{H}$, the complete dual-stream warping operator can be summarized as

$$\left(\widetilde{\mathbf{V}}^{i+1}, \widetilde{\mathbf{D}}^{i+1}, \widetilde{\mathbf{M}}^{i+1}\right) = \mathcal{R}_{\text{dual}}\left(\mathcal{M}_i, \mathbf{C}^{i+1}\right) = \mathcal{H}\Big(\mathcal{R}\left(\mathcal{M}_i^{\text{app}}, \mathbf{C}^{i+1}\right), \mathcal{R}\left(\mathcal{M}_i^{\text{geo}}, \mathbf{C}^{i+1}\right)\Big). \tag{13}$$

The fused conditions $(\widetilde{\mathbf{V}}^{i+1}, \widetilde{\mathbf{D}}^{i+1}, \widetilde{\mathbf{M}}^{i+1})$ can be subsequently passed to $\mathcal{F}_{\text{app}}$ for the next autoregressive generation step. This pixel-wise selection preserves the sharp appearance produced by

selective warping over most regions while exploiting dense geometry where depth disagreement indicates a potential occlusion conflict. In this way, dual-stream warping suppresses both background leakage under large viewpoint changes and appearance degradation over successive iterations. Notably, while formulated as two logical memory states, the two streams share the same accumulated RGB-D observations, camera poses, and per-trajectory provenance masks. In practice, they are implemented through two warping passes with different source-mask configurations: the appearance pass respects the stored provenance masks, whereas the geometry pass treats all source pixels as valid. Thus, no duplicated point-cloud memory is required. Kindly refer to more details in Appendix B.4.

3.4 Geometry-aware Video Refinement

With the novel-view RGB video $\mathbf{V}^i$, its dense depth video $\mathbf{D}^i$, and the memory update mechanism described above, the core autoregressive loop is already sufficient to support the generation of subsequent target trajectories. However, we observe that $\mathbf{V}^i$ may occasionally exhibit subtle texture flicker and high-frequency noise, which degrade its perceptual quality (as shown in Fig. 4(c)). Fortunately, our video depth completion model remains robust to these mild appearance artifacts and produces temporally stable and spatially smooth depth predictions. This observation motivates an optional geometry-aware refinement stage that leverages the recovered depth as a reliable structural prior to further improve the generated appearance.

Specifically, we introduce a diffusion-based, geometry-aware video refinement model $\mathcal{F}_{\text{ref}}$, which takes the partial RGB rendering $\overline{\mathbf{V}}^i$ and the completed target-view depth $\mathbf{D}^i$ as inputs:

$$\mathbf{V}^i_{\text{ref}} = \mathcal{F}_{\text{ref}}\left(\overline{\mathbf{V}}^i, \mathbf{D}^i\right), \tag{14}$$

where $\mathbf{V}^i_{\text{ref}} = \{\mathbf{I}^i_{\text{ref},t}\}_{t=1}^T$ denotes the refined novel-view video. The partial RGB rendering provides appearance guidance from the 4D memory, while the dense depth supplies complete and temporally coherent geometric guidance across the target views. Conditioned on these complementary signals, $\mathcal{F}_{\text{ref}}$ reduces texture flicker and noise while preserving the recovered scene structure, yielding refined videos with improved visual fidelity and temporal coherence. As an optional module, this refinement stage enhances the final video quality without affecting the core autoregressive world-expansion process. More details are provided in Appendix B.5.

4 Experiments

4.1 Experimental setups

Implementation Details. All diffusion-based stages of our framework share Wan2.1-1.3B Wan et al. (2025) as the generative backbone. The models operate on 81-frame video clips at a resolution of 480×832. Our training corpus comprises 236K samples, including 136K samples derived from the MultiCamVideo dataset Bai et al. (2025) and 100K samples from the OpenVid dataset Nan et al. (2025). All experiments are conducted using eight NVIDIA H20 GPUs. For systematic evaluation under unconstrained real-world scenarios, we establish a benchmark comprising 128 diverse in-the-wild monocular videos. For each input sequence, we synthesize novel views along a fixed suite of 14 camera trajectories: eight azimuthal trajectories at $\{\pm10^\circ, \pm20^\circ, \pm30^\circ, \pm40^\circ\}$, four elevation trajectories at $\{\pm10^\circ, \pm20^\circ\}$, and two radial zoom-out trajectories with normalized displacements of 0.5 and 1.0. The orbital radius is fixed to unit length for all rotational trajectories. Metrics are computed over all $128 \times 14 = \mathbf{1{,}792}$ generated videos. See more details in Appendix C.

Table 1: **Quantitative comparison of generated novel-view videos.** Demiurge achieves the best perceptual quality, cross-trajectory consistency, and camera controllability. Red, orange, and yellow denote the best, second-best, and third-best results, respectively.

Method	FVD ↓	Q-Align-Video ↑		Q-Align-Image ↑		P-MSE ↓	R-Err ↓	T-Err ↓	View Sync ↑
		Quality	Aesthetics	Quality	Aesthetics				
TrajCrafter Yu et al. (2025b)	597.652	4.262	3.242	3.950	3.035	0.007	4.106	2.403	21.426
InspatioWorld Team et al. (2026)	369.577	4.413	3.366	4.196	3.135	0.017	7.119	2.159	40.607
ReCamMaster Bai et al. (2025)	478.897	4.189	3.162	3.956	2.966	0.046	4.248	2.934	39.624
Gen3C Ren et al. (2025)	363.400	4.277	3.266	4.060	3.074	0.008	6.645	2.647	46.562
Voyager Huang et al. (2025)	839.415	4.081	3.141	3.774	2.950	0.031	10.803	3.715	22.106
Spatia Zhao et al. (2026)	477.889	4.234	3.347	4.093	3.125	0.023	6.798	3.172	33.171
Ours	334.518	4.549	3.507	4.331	3.291	0.001	3.655	1.666	64.893

Table 2: **Quantitative comparison of 3DGS reconstructed from generated videos.** Average is the geometric mean of MSE $= 10^{-\mathrm{PSNR}/10}$, $\sqrt{1-\mathrm{SSIM}}$, and LPIPS. Red, orange, and yellow denote the best, second-best, and third-best results, respectively.

Method	FVD ↓	Q-Align-Video ↑		Q-Align-Image ↑		PSNR ↑	SSIM ↑	LPIPS ↓	Average ↓
		Quality	Aesthetics	Quality	Aesthetics				
TrajCrafter Yu et al. (2025b)	1258.756	3.430	2.703	2.837	2.481	20.153	0.671	0.368	0.127
InspatioWorld Team et al. (2026)	1141.013	3.590	2.747	3.098	2.5213	21.767	0.706	0.296	0.102
ReCamMaster Bai et al. (2025)	1318.239	3.343	2.547	2.810	2.322	22.212	0.752	0.327	0.099
Gen3C Ren et al. (2025)	1143.709	3.390	2.623	2.865	2.399	22.161	0.727	0.282	0.096
Voyager Huang et al. (2025)	1677.240	2.970	2.385	2.263	2.174	16.116	0.568	0.552	0.207
Spatia Zhao et al. (2026)	1217.144	3.485	2.726	2.922	2.515	21.496	0.706	0.343	0.110
Ours	1128.544	3.625	2.836	3.146	2.617	24.360	0.769	0.237	0.075

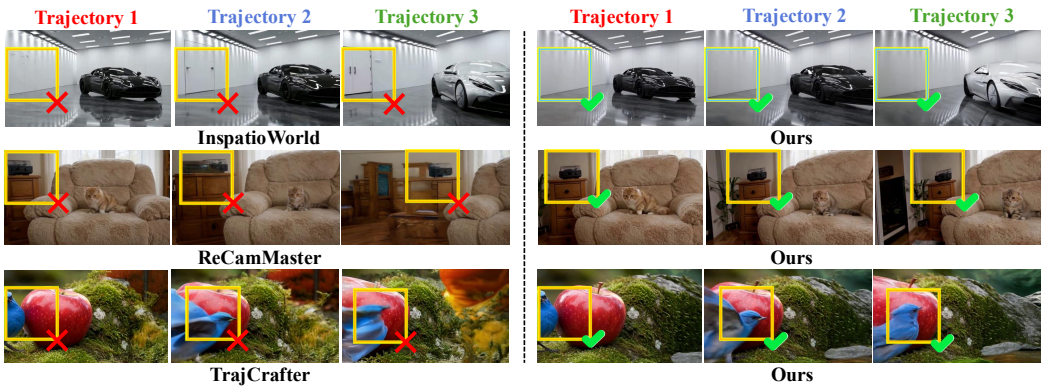


Figure 3: **Qualitative comparison.** Demiurge better preserves appearance and geometry across camera trajectories, reducing hallucinations, content drift, and structural distortions.

Baselines and metrics. We compare our method against six representative camera-controlled video generation approaches: TrajCrafter Yu et al. (2025b), InSpatioWorld Team et al. (2026), ReCamMaster Bai et al. (2025), Gen3C Ren et al. (2025), Voyager Huang et al. (2025), and Spatia Zhao et al. (2026). We evaluate all methods along two complementary dimensions: video quality and consistency. *Video Quality.* We report FVD Unterthiner et al. (2018) and Q-Align Wu et al. (2023) to assess the perceptual quality and aesthetics of generated videos. *Consistency.* We introduce Projection MSE (P-MSE) to measure input–output consistency by projecting the input video onto the generated camera trajectory and computing the MSE over valid overlapping regions. We further use PSNR Huynh-Thu & Ghanbari (2008), SSIM Wang et al. (2004), and LPIPS Zhang et al. (2018) to assess the rendering quality of 3DGS representations reconstructed from the generated multi-view videos, and View Sync Edstedt et al. (2024) to quantify cross-view correspondence by counting matched pixels above a confidence threshold. Finally, we report rotation error (R-err) and translation error (T-err) He et al. (2024) between the recovered and target camera trajectories to evaluate trajectory adherence. We provide more details in Appendix D.

4.2 Main Results

As shown in Tab. 1, we first evaluate the quality and consistency of the generated novel-view videos. Demiurge achieves the best results across all reported metrics, obtaining the lowest FVD of 334.518 and the highest quality and aesthetics scores under both Q-Align-Video and Q-Align-Image. Compared with the strongest baselines, it reduces P-MSE by 85.7% and improves View Sync by 39.4%, demonstrating substantially better content preservation and cross-trajectory consistency. It further reduces R-err and T-err by 11.0% and 22.8%, respectively, confirming more accurate adherence to the target camera trajectories. We additionally reconstruct a 3D Gaussian Splatting (3DGS) representation from the videos generated by each method and evaluate its rendering quality in Tab. 2. Demiurge again performs best across all metrics, achieving a PSNR of 24.360, an SSIM of 0.769, and an LPIPS of 0.237. In particular, it improves PSNR by 2.148 dB and reduces LPIPS and the aggregate reconstruction error by 16.0% and 21.9%, respectively, indicating that its generated views form a more coherent 3D representation. The qualitative comparisons in Fig. 3 further show that

Table 3: **Component ablation of Demiurge.** NM, VDC, DSW, and Ref. denote noise monitoring, video depth completion, dual-stream warping, and geometry-aware refinement, respectively. *M. Est.* directly uses monocular video depth estimation.

Variant	Components				Generated Videos				Reconstructed 3DGS			
	NM	Depth Module	DSW	Ref.	FVD↓	Q-Align-V↑	P-MSE↓	View Sync↑	PSNR↑	SSIM↑	LPIPS↓	Avg.↓
w/o Noise Monitor	–	VDC.	✓	✓	339.238	4.460	0.0013	62.575	24.217	0.757	0.239	0.076
w/ Monocular Depth	✓	M. Est.	✓	✓	388.597	4.447	0.0020	54.226	24.132	0.749	0.255	0.079
w/o Dual-stream Warping	✓	VDC.	–	✓	383.525	4.412	0.0049	51.498	23.886	0.744	0.254	0.081
w/o Refinement	✓	VDC.	✓	–	347.922	4.521	0.0042	60.680	24.325	0.756	0.239	0.076
Full model	✓	VDC.	✓	✓	334.518	4.549	0.0010	64.893	24.360	0.769	0.237	0.075

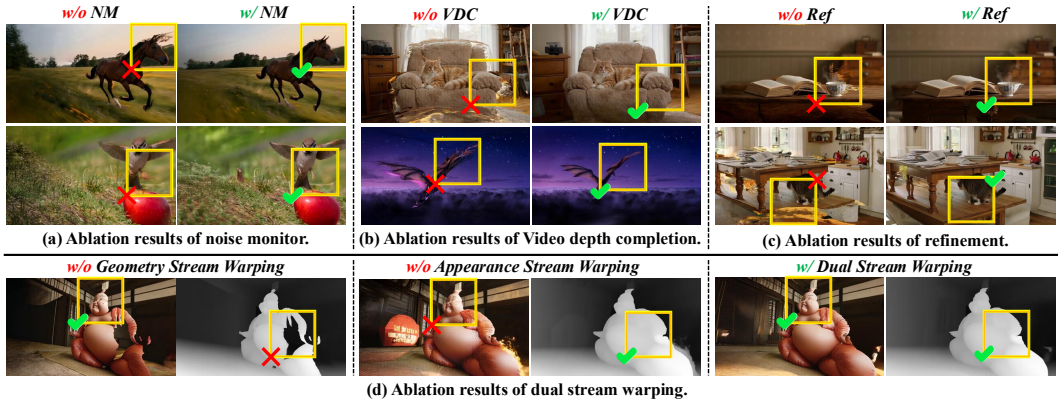


Figure 4: **Qualitative component ablation.** NM suppresses projection artifacts, VDC improves geometric consistency, dual-stream warping preserves both appearance and occlusion geometry, and refinement removes residual visual artifacts.

competing methods often introduce viewpoint-dependent hallucinations, appearance changes, and geometric distortions, whereas Demiurge better preserves object identity, fine-grained appearance, and spatial layout across diverse viewpoints. Together, these results demonstrate that our proposed method enables high-quality video generation while maintaining long-range consistency across camera trajectories and supporting geometrically coherent 3DGS reconstruction. Due to space constraints, additional results on key capabilities, qualitative comparisons, compatibility with existing methods, and visualizations are provided in Appendices E, F, H, and I.

4.3 Ablation Studies

Tab. 3 and Fig. 4 validate the complementary roles of our components. The noise monitor suppresses unreliable memory projections and the resulting boundary artifacts. Replacing video depth completion with monocular depth estimation increases P-MSE from 0.0010 to 0.0020 and decreases View Sync from 64.893 to 54.226, demonstrating the importance of memory-consistent geometry recovery. Dual-stream warping contributes most to cross-trajectory consistency: removing it raises P-MSE to 0.0049 and reduces View Sync to 51.498. As visualized in Fig. 4(d), its appearance and geometry streams respectively preserve visual details and reliable occlusion geometry. Finally, geometry-aware refinement further removes residual artifacts and improves local details. Together, these components yield high-quality generation with consistent appearance and geometry. Additional visualizations are provided in Appendix G.

5 Conclusion

We introduced ***Demiurge***, an autoregressive video rerendering framework for constructing navigable 4D worlds from monocular videos. Unlike existing methods that synthesize each camera trajectory independently, Demiurge continuously consolidates generated appearance and recovered geometry into an evolving 4D point-cloud memory through a generate–recover–update cycle. This explicit memory enables reliable reuse of historical observations and preserves long-range consistency across camera trajectories. Extensive experiments establish state-of-the-art performance in perceptual quality, trajectory accuracy, and cross-trajectory consistency, as well as demonstrating robust large-parallax rerendering and geometry-faithful 3DGS reconstruction.

AI USE STATEMENT

Generative AI tools were used solely for grammar correction and linguistic refinement of author-written text. They were not used for research conceptualization, method development, code generation, experimental design, data analysis, or the formulation of scientific claims. All AI-assisted edits were carefully reviewed by the authors to preserve the intended technical meaning. The authors take full responsibility for the accuracy and integrity of the final manuscript.

ETHICS STATEMENT

Demiurge is developed for research on controllable video rerendering and 4D scene representation. As with other generative video methods, it may be misused to produce misleading content or process identifiable individuals without consent, and it may inherit biases from pretrained models and training data. We encourage responsible use, disclosure of generated content, respect for privacy and data licenses, and the adoption of provenance and watermarking mechanisms in practical deployments.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training and inference procedures, 4D memory construction, depth alignment, dual-stream warping, and evaluation protocols in the main paper and supplementary material. The supplementary material further specifies the metric implementations, runtime settings, and additional qualitative and ablation results. Training data composition, model backbones, video resolution, sequence length, and computational resources are also reported. Additional videos and supporting materials are available on our anonymous project page. We intend to release the source code, model checkpoints, and evaluation scripts to facilitate future reproduction and comparison.

REFERENCES

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. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22875–22889. IEEE, 2025a.

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

Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, et al. Recammaster: Camera-controlled generative rendering from a single video. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14834–14844. IEEE, 2025.

Shengqu Cai, Ceyuan Yang, Lvmin Zhang, Yuwei Guo, Junfei Xiao, Ziyan Yang, Yinghao Xu, Zhenheng Yang, Alan Yuille, Leonidas Guibas, et al. Mixture of contexts for long video generation. In *International Conference on Learning Representations*, volume 2026, pp. 92075–92093, 2026.

Junyi Chen, Haoyi Zhu, Xianglong He, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Yang Zhou, Zizun Li, Zhoujie Fu, Jiangmiao Pang, et al. Deepverse: 4d autoregressive video generation as a world model. *arXiv preprint arXiv:2506.01103*, 2025a.

Luxi Chen, Zihan Zhou, Min Zhao, Yikai Wang, Ge Zhang, Wenhao Huang, Hao Sun, Ji-Rong Wen, and Chongxuan Li. Flexworld: Progressively expanding 3d scenes for flexiable-view synthesis. *arXiv preprint arXiv:2503.13265*, 2025b.

Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19790–19800. IEEE, 2024.

Xiao Fu, Shitao Tang, Min Shi, Xian Liu, Jinwei Gu, Ming-Yu Liu, Dahua Lin, and Chen-Hsuan Lin. Plenoptic video generation. *arXiv preprint arXiv:2601.05239*, 2026.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, et al. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–12, 2025.

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

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

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. Depthcrafter: Generating consistent long depth sequences for open-world videos. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2005–2015. IEEE, 2025.

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.

Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):1, 2008.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

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*, 2(3):6, 2025a.

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

Kuan Heng Lin, Zhizheng Liu, Pablo Salamanca, Yash Kant, Ryan Burgert, Yuancheng Xu, Koichi Namekata, Yiwei Zhao, Bolei Zhou, Micah Goldblum, et al. Vista4d: Video reshooting with 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32671–32682, 2026.

Zhiheng Liu, Xueqing Deng, Shoufa Chen, Angtian Wang, Qiushan Guo, Mingfei Han, Zeyue Xue, Mengzhao Chen, Ping Luo, and Linjie Yang. Worldweaver: Generating long-horizon video worlds via rich perception. *Advances in Neural Information Processing Systems*, 38:46268–46293, 2026.

Yue Ma, Kunyu Feng, Zhongyuan Hu, Xinyu Wang, Yucheng Wang, Mingzhe Zheng, Bingyuan Wang, Qinghe Wang, Xuanhua He, Hongfa Wang, et al. Controllable video generation: A survey. *arXiv preprint arXiv:2507.16869*, 2025.

Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model. *arXiv preprint arXiv:2507.17744*, 2025.

Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *International conference on learning representations*, volume 2025, pp. 1045–1064, 2025.

Jangho Park, Taesung Kwon, and Jong Chul Ye. Zero4d: Training-free 4d video generation from single video using off-the-shelf video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4045–4054, 2026.

Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 577–582. 2023.

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

Manuel-Andreas Schneider, Lukas Höllein, and Matthias Nießner. Worldexplorer: Towards generating fully navigable 3d scenes. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, 2025.

Chenxi Song, Yanming Yang, Tong Zhao, Ruibo Li, and Chi Zhang. Taming video models for 3d and 4d generation via zero-shot camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40352–40363, 2026.

InSpatio Team, Donghui Shen, Guofeng Zhang, Haomin Liu, Haoyu Ji, Hujun Bao, Hongjia Zhai, Jialin Liu, Jing Guo, Nan Wang, et al. Inspatio-world: A real-time 4d world simulator via spatiotemporal autoregressive modeling. *arXiv preprint arXiv:2604.07209*, 2026.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Peiyan Tu, Hanxin Zhu, Jingwen Sun, Shaojie Ren, Cong Wang, Yuyan Xu, Jiayi Luo, Xiaoqian Cheng, and Zhibo Chen. Embody4d: A generalist data engine for embodied 4d world modeling. *arXiv preprint arXiv:2605.01799*, 2026.

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

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

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

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*, pp. 1–11, 2024.

Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Liang Liao, Chunyi Li, Yixuan Gao, Annan Wang, Erli Zhang, Wenxiu Sun, et al. Q-align: Teaching lmms for visual scoring via discrete text-defined levels. *arXiv preprint arXiv:2312.17090*, 2023.

Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. *Advances in Neural Information Processing Systems*, 38:49371–49393, 2026.

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

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

Yuxue Yang, Lue Fan, Ziqi Shi, Junran Peng, Feng Wang, and Zhaoxiang Zhang. Neoverse: Enhancing 4d world model with in-the-wild monocular videos. *arXiv preprint arXiv:2601.00393*, 2026.

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*, pp. 1–11, 2025a.

Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 100–111. IEEE, 2025b.

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.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Jinjing Zhao, Fangyun Wei, Zhening Liu, Hongyang Zhang, Chang Xu, and Yan Lu. Spatia: Video generation with updatable spatial memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4245–4257, 2026.

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

Sixiao Zheng, Minghao Yin, Wenbo Hu, Xiaoyu Li, Ying Shan, and Yanwei Fu. Versecrafter: Dynamic realistic video world model with 4d geometric control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40277–40290, 2026.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024.

Hanxin Zhu, Cong Wang, Peiyan Tu, Jiayi Luo, Tianyu He, Xin Jin, and Zhibo Chen. Gta: Advancing image-to-3d world generation via geometry then appearance video diffusion. *International Journal of Computer Vision*, 134(9):410, 2026.

APPENDIX FOR DEMIURGE: FORGING NAVIGABLE 4D WORLDS VIA AUTOREGRESSIVE VIDEO RERENDERING

A 4D Point-Cloud Memory **15**

A.1 4D Point-Cloud Representation 15

A.2 Point-Cloud Warping 15

B Additional Method Details **17**

B.1 Noise Monitor 17

B.2 Cache-Guided Appearance Video Diffusion 18

B.3 Cache-Consistent Video Depth Completion 18

B.4 Dual-Stream Warping 21

B.5 Geometry-Aware Video Refinement 22

C Training and Inference Details **22**

C.1 Training Data Construction 22

C.2 Training Configurations 24

C.3 Inference Configurations 24

D Evaluation Details **24**

D.1 Metric Definitions 24

E Key capabilities **27**

E.1 Coherent spatiotemporal lattice generation 27

E.2 Large-parallax video rerendering 27

E.3 Geometry-faithful outputs for 3DGS reconstruction 27

F More qualitative comparisons **27**

G Additional ablation results **28**

H Compatibility with Existing Method **28**

I Additional visualizations **28**

J Runtime Analysis **28**

K Limitations and future work **29**

A 4D POINT-CLOUD MEMORY

A.1 4D POINT-CLOUD REPRESENTATION

As introduced in the main paper, Demiurge represents a dynamic scene using an evolving 4D point-cloud memory $\mathcal{M}_i = \{\mathcal{P}_{i,t}\}_{t=1}^{T}$, where $\mathcal{P}_{i,t}$ denotes the colored 3D point cloud accumulated at time t after processing the i-th target trajectory. Each $\mathcal{P}_{i,t}$ represents an instantaneous 3D snapshot of the scene, while their temporally ordered collection constitutes the complete 4D representation.

Let $\mathbf{D}^0 = \{\mathbf{D}_t^0\}_{t=1}^{T}$ denote the depth video estimated from the input video $\mathbf{V}^0 = \{\mathbf{I}_t^0\}_{t=1}^{T}$. For a pixel $\mathbf{p} = (u, v)$, we define its homogeneous image coordinate as

$$\overline{\mathbf{p}} = [u \quad v \quad 1]^{\mathsf{T}}. \tag{15}$$

Following the camera convention used by our warping implementation, a source-camera point is transformed into the shared world coordinate system by the inverse camera transformation. Therefore, the world-space position associated with pixel $\mathbf{p}$ in the t-th input frame is

$$\begin{bmatrix} \mathbf{X}_t^0(\mathbf{p}) \\ 1 \end{bmatrix} = \left(\mathbf{C}_t^0\right)^{-1} \begin{bmatrix} \mathbf{D}_t^0(\mathbf{p})\mathbf{K}^{-1}\overline{\mathbf{p}} \\ 1 \end{bmatrix}, \tag{16}$$

where $\mathbf{K}$ is the camera intrinsic matrix. The initial colored point cloud at time t is then constructed as

$$\mathcal{P}_{0,t} = \left\{ \left(\mathbf{X}_t^0(\mathbf{p}), \mathbf{I}_t^0(\mathbf{p})\right) \;\middle|\; \mathbf{p} \in \Omega_t^0 \right\}, \tag{17}$$

where Ω_t^0 denotes the set of valid pixels in the input RGB-D frame. The initial 4D memory is consequently

$$\mathcal{M}_0 = \{\mathcal{P}_{0,t}\}_{t=1}^{T}. \tag{18}$$

After generating the i-th novel-view video $\mathbf{V}^i = \{\mathbf{I}_t^i\}_{t=1}^{T}$ and recovering its dense depth video $\mathbf{D}^i = \{\mathbf{D}_t^i\}_{t=1}^{T}$, we similarly back-project its valid RGB-D observations. For every $\mathbf{p} \in \Omega_t^i$, the corresponding world-space point is

$$\begin{bmatrix} \mathbf{X}_t^i(\mathbf{p}) \\ 1 \end{bmatrix} = \left(\mathbf{C}_t^i\right)^{-1} \begin{bmatrix} \mathbf{D}_t^i(\mathbf{p})\mathbf{K}^{-1}\overline{\mathbf{p}} \\ 1 \end{bmatrix}, \tag{19}$$

where Ω_t^i denotes the set of valid pixels selected by the memory-update procedure. These colored points are accumulated into the point-cloud snapshot at the same timestamp:

$$\mathcal{P}_{i,t} = \mathcal{P}_{i-1,t} \uplus \left\{ \left(\mathbf{X}_t^i(\mathbf{p}), \mathbf{I}_t^i(\mathbf{p})\right) \;\middle|\; \mathbf{p} \in \Omega_t^i \right\}, \tag{20}$$

where $\uplus$ denotes observation accumulation without geometric averaging or point merging. In practice, we retain the RGB-D observations, camera poses, and validity masks associated with individual trajectories, and instantiate their corresponding colored 3D points on demand during rendering.

Importantly, an RGB-D observation at time t is accumulated only into $\mathcal{P}_{i,t}$ and never into $\mathcal{P}_{i,t'}$ for $t' \neq t$. Consequently, a moving object is represented at different 3D locations in the corresponding point-cloud snapshots over time, rather than being duplicated within a single static point cloud.

A.2 POINT-CLOUD WARPING

Although the memory is conceptually represented as a sequence of colored point clouds, its factorized implementation allows us to render each previously observed RGB-D trajectory independently and then composite the warped results. Consider rendering the target frame at time t along the i-th trajectory. Let $\mathbf{I}_t^j$, $\mathbf{D}_t^j$, $\mathbf{M}_t^j$, and $\mathbf{C}_t^j$ denote the RGB frame, depth map, validity mask, and camera pose associated with a previously observed trajectory $j \in \{0, \ldots, i-1\}$.

For a source pixel $\mathbf{p}$, its source-camera 3D position is

$$\mathbf{x}_{j,t}(\mathbf{p}) = \mathbf{D}_t^j(\mathbf{p})\mathbf{K}^{-1}\overline{\mathbf{p}}. \tag{21}$$

Following the implementation, the relative transformation from the source camera $\mathbf{C}_t^j$ to the target camera $\mathbf{C}_t^i$ is

$$\mathbf{T}_{j\to i,t} = \mathbf{C}_t^i \left(\mathbf{C}_t^j\right)^{-1}. \tag{22}$$

The source point is transformed into the target camera coordinate system according to

$$\begin{bmatrix} \mathbf{x}_{j\to i,t}(\mathbf{p}) \\ 1 \end{bmatrix} = \mathbf{T}_{j\to i,t} \begin{bmatrix} \mathbf{x}_{j,t}(\mathbf{p}) \\ 1 \end{bmatrix}, \tag{23}$$

where

$$\mathbf{x}_{j\to i,t}(\mathbf{p}) = \begin{bmatrix} x_{j\to i,t}(\mathbf{p}) \\ y_{j\to i,t}(\mathbf{p}) \\ z_{j\to i,t}(\mathbf{p}) \end{bmatrix}. \tag{24}$$

Its homogeneous coordinate on the target image plane is

$$\overline{\mathbf{p}}_{j\to i,t} = \mathbf{K}\mathbf{x}_{j\to i,t}(\mathbf{p}), \tag{25}$$

and the corresponding continuous pixel coordinate is

$$\mathbf{p}_{j\to i,t} = \begin{bmatrix} \overline{p}_x/\overline{p}_z \\ \overline{p}_y/\overline{p}_z \end{bmatrix}. \tag{26}$$

The depth of this point in the target camera coordinate system is

$$z_{j\to i,t}(\mathbf{p}) = [\mathbf{x}_{j\to i,t}(\mathbf{p})]_3 . \tag{27}$$

Depth-aware bilinear splatting. Since $\mathbf{p}_{j\to i,t}(\mathbf{p})$ generally lies at a non-integer image coordinate, we distribute its RGB and depth values to the four neighboring target pixels through bilinear splatting. Let $\mathcal{N}(\mathbf{p}_{j\to i,t}(\mathbf{p}))$ denote these four neighboring pixels. For an integer target pixel $\mathbf{q} = (q_x, q_y)$ in this neighborhood, the bilinear proximity weight is

$$b_{j,t}(\mathbf{p},\mathbf{q}) = \left(1 - \left|p^x_{j\to i,t}(\mathbf{p}) - q_x\right|\right)\left(1 - \left|p^y_{j\to i,t}(\mathbf{p}) - q_y\right|\right). \tag{28}$$

We additionally apply depth-aware weighting to suppress contributions from distant points when multiple source pixels are splatted onto the same target neighborhood. Specifically, we define

$$\ell_{j,t}(\mathbf{p}) = \log\left(1 + \mathrm{clip}\left(z_{j\to i,t}(\mathbf{p}), 0, 1000\right)\right). \tag{29}$$

Let $\mathcal{B}$ denote the collection of source pixels included in the current batched warping operation. The normalization factor used by the implementation is

$$\ell^{\max}_{\mathcal{B}} = \max_{(j,t,\mathbf{p})\in\mathcal{B}} \ell_{j,t}(\mathbf{p}). \tag{30}$$

The complete splatting weight is then

$$w_{j,t}(\mathbf{p},\mathbf{q}) = b_{j,t}(\mathbf{p},\mathbf{q})\,\mathbf{M}^j_t(\mathbf{p})\,\exp\left(-50\frac{\ell_{j,t}(\mathbf{p})}{\ell^{\max}_{\mathcal{B}}}\right). \tag{31}$$

For each source trajectory j, the accumulated splatting weight at target pixel $\mathbf{q}$ is

$$s^i_{j,t}(\mathbf{q}) = \sum_{\mathbf{p}:\,\mathbf{q}\in\mathcal{N}(\mathbf{p}_{j\to i,t}(\mathbf{p}))} w_{j,t}(\mathbf{p},\mathbf{q}). \tag{32}$$

For target pixels satisfying $s^i_{j,t}(\mathbf{q}) > 0$, the RGB and depth warped from source trajectory j are computed as

$$\mathbf{I}_{j\to i,t}(\mathbf{q}) = \frac{\sum_{\mathbf{p}:\,\mathbf{q}\in\mathcal{N}(\mathbf{p}_{j\to i,t}(\mathbf{p}))} w_{j,t}(\mathbf{p},\mathbf{q})\mathbf{I}^j_t(\mathbf{p})}{s^i_{j,t}(\mathbf{q})}, \tag{33}$$

and

$$\mathbf{D}_{j\to i,t}(\mathbf{q}) = \frac{\sum_{\mathbf{p}:\,\mathbf{q}\in\mathcal{N}(\mathbf{p}_{j\to i,t}(\mathbf{p}))} w_{j,t}(\mathbf{p},\mathbf{q})z_{j\to i,t}(\mathbf{p})}{s^i_{j,t}(\mathbf{q})}. \tag{34}$$

The corresponding warped validity mask is

$$\mathbf{M}_{j\to i,t}(\mathbf{q}) = \mathbb{I}\left[s^i_{j,t}(\mathbf{q}) > 0\right]. \tag{35}$$

Cross-trajectory z-buffer composition. After independently warping all previously observed trajectories, we combine their results using a nearest-depth z-buffer. For each target pixel $\mathbf{q}$, the set of valid warped sources is

$$\mathcal{A}_t^i(\mathbf{q}) = \{j \mid \mathbf{M}_{j\to i,t}(\mathbf{q}) > 0.5, \quad \mathbf{D}_{j\to i,t}(\mathbf{q}) > \tau_0\}, \tag{36}$$

where $\tau_0 = 10^{-5}$. If multiple warped sources cover the same target pixel, we select the source with the smallest warped depth:

$$j_t^{i,*}(\mathbf{q}) = \underset{j\in\mathcal{A}_t^i(\mathbf{q})}{\arg\min}\, \mathbf{D}_{j\to i,t}(\mathbf{q}). \tag{37}$$

The resulting point-cloud rendering at target pixel $\mathbf{q}$ is defined as

$$\left(\mathbf{I}_{\mathrm{pc},t}^i(\mathbf{q}), \mathbf{D}_{\mathrm{pc},t}^i(\mathbf{q}), \mathbf{M}_{\mathrm{pc},t}^i(\mathbf{q})\right) = \begin{cases} \left(\mathbf{I}_{j_t^{i,*}(\mathbf{q})\to i,t}(\mathbf{q}), \mathbf{D}_{j_t^{i,*}(\mathbf{q})\to i,t}(\mathbf{q}), 1\right), & \mathcal{A}_t^i(\mathbf{q}) \neq \varnothing, \\ (\mathbf{0}, 0, 0), & \mathcal{A}_t^i(\mathbf{q}) = \varnothing. \end{cases} \tag{38}$$

Accordingly, the elementary point-cloud rendering operator is

$$\left(\mathbf{I}_{\mathrm{pc},t}^i, \mathbf{D}_{\mathrm{pc},t}^i, \mathbf{M}_{\mathrm{pc},t}^i\right) = \mathcal{R}\left(\mathcal{P}_{i-1,t}, \mathbf{C}_t^i\right), \tag{39}$$

where the subscript pc denotes the RGB, depth, and visibility outputs of point-cloud rendering, $\mathcal{R}$ consists of the depth-aware bilinear splatting defined in Eq. 21– Eq. 35, followed by the cross-trajectory z-buffer composition defined in Eq. 36– Eq. 38. Equivalently,

$$\mathcal{R}\left(\mathcal{P}_{i-1,t}, \mathbf{C}_t^i\right) = \mathrm{ZBuf}\left(\left\{\mathcal{W}\left(\mathbf{I}_t^j, \mathbf{D}_t^j, \mathbf{M}_t^j, \mathbf{C}_t^j, \mathbf{C}_t^i, \mathbf{K}\right)\right\}_{j=0}^{i-1}\right), \tag{40}$$

where $\mathcal{W}$ denotes the depth-aware bilinear forward-warping operation for an individual historical trajectory.

The rendering operation is applied independently at every timestamp. In particular, rendering the target frame at time t involves only $\mathcal{P}_{i-1,t}$ and $\mathbf{C}_t^i$. No point from $\mathcal{P}_{i-1,t'}$, where $t' \neq t$, participates in the rendering. Dynamic content is therefore represented by its evolving geometry across temporally indexed point-cloud snapshots, allowing static backgrounds and moving objects to be handled uniformly without explicit dynamic–static decomposition.

B Additional Method Details

B.1 Noise Monitor

We provide the implementation details of the non-learned noise monitor introduced in the main paper. Given a rendered depth frame, we identify unreliable pixels using three complementary criteria: invalid depth, large depth gradients, and high local depth variance. We set the corresponding thresholds to

$$\tau_0 = 10^{-5}, \qquad \tau_g = 0.03, \qquad \tau_v = 10^{-4}. \tag{41}$$

The horizontal and vertical depth derivatives are computed using finite differences, and their magnitude is given by

$$G_t^i(\mathbf{p}) = \sqrt{\left(\nabla_x \widetilde{\mathbf{D}}_t^i(\mathbf{p})\right)^2 + \left(\nabla_y \widetilde{\mathbf{D}}_t^i(\mathbf{p})\right)^2}. \tag{42}$$

The local depth variance is computed over a 3×3 neighborhood using

$$S_t^i(\mathbf{p}) = \mathrm{Mean}_{\mathcal{N}_3(\mathbf{p})}\left[\left(\widetilde{\mathbf{D}}_t^i\right)^2\right] - \left(\mathrm{Mean}_{\mathcal{N}_3(\mathbf{p})}\left[\widetilde{\mathbf{D}}_t^i\right]\right)^2. \tag{43}$$

A pixel is marked as unreliable if it satisfies any of the three criteria. The resulting noise mask is simultaneously applied to the rendered RGB, depth, and visibility conditions, ensuring that the video diffusion model does not receive mutually inconsistent conditioning signals. In our default setting, the cleaning operation is applied once after point-cloud rendering. The monitor is fully deterministic, introduces no learnable parameters, and incurs negligible computational overhead.

B.2 Cache-Guided Appearance Video Diffusion

Our cache-guided appearance model $\mathcal{F}_{\text{app}}$ is initialized from the pretrained Wan2.1-Fun-1.3B-InP video diffusion model. As described in the main paper, it takes the cleaned partial RGB video $\overline{\mathbf{V}}^i$, partial depth video $\overline{\mathbf{D}}^i$, and visibility mask $\overline{\mathbf{M}}^i$ as conditions and synthesizes the novel-view video $\mathbf{V}^i$ along the target camera trajectory.

Latent condition encoding. We first normalize the three conditioning videos to the input range of the Wan VAE. Since the depth and visibility conditions are single-channel signals, each of them is replicated over three channels before VAE encoding. Let $\mathcal{E}_{\text{vae}}$ denote the frozen Wan VAE encoder. The condition latents are given by

$$\begin{aligned} \mathbf{z}^i_{\text{rgb}} &= \mathcal{E}_{\text{vae}}\left(\overline{\mathbf{V}}^i\right), \\ \mathbf{z}^i_{\text{dep}} &= \mathcal{E}_{\text{vae}}\left(\text{Rep}_3\left(\overline{\mathbf{D}}^i\right)\right), \\ \mathbf{z}^i_{\text{mask}} &= \mathcal{E}_{\text{vae}}\left(\text{Rep}_3\left(\overline{\mathbf{M}}^i\right)\right), \end{aligned} \tag{44}$$

where $\text{Rep}_3(\cdot)$ replicates a single-channel video over the RGB channels. All three conditions are therefore encoded into the same spatio-temporal latent space as the target video.

Latent-space condition injection. Let $\mathbf{z}^i_\tau$ denote the noisy target RGB latent at diffusion timestep τ. We inject the cache-derived conditions through channel-wise latent concatenation:

$$\mathbf{h}^i_\tau = \text{Concat}_c\left[\mathbf{z}^i_\tau, \mathbf{z}^i_{\text{rgb}}, \mathbf{z}^i_{\text{dep}}, \mathbf{z}^i_{\text{mask}}\right], \tag{45}$$

where Concat_c denotes concatenation along the channel dimension. The Wan VAE produces a 16-channel latent for each video. Consequently, $\mathbf{h}^i_\tau$ contains 64 channels, including 16 noisy target channels and 48 conditioning channels.

To accommodate this expanded input, we replace the original input projection with a 3D convolutional patch embedding $\mathcal{P}_{\text{app}}$ that maps the 64-channel latent volume to the hidden dimension of the pretrained Wan DiT:

$$\mathbf{x}^i_\tau = \mathcal{P}_{\text{app}}\left(\mathbf{h}^i_\tau\right). \tag{46}$$

The kernel size and stride of $\mathcal{P}_{\text{app}}$ follow the original spatio-temporal patchification configuration of the Wan backbone.

Video diffusion backbone. The resulting tokens are processed by the original Wan DiT blocks. Each block consists of spatio-temporal self-attention, text cross-attention, and a feed-forward network. Three-dimensional rotary position embeddings are applied along the temporal, height, and width axes, while the diffusion timestep modulates the self-attention and feed-forward layers. The caption of the input video is encoded by the UMT5 text encoder and injected into all DiT blocks through cross-attention.

B.3 Cache-Consistent Video Depth Completion

We provide additional architectural details of the cache-consistent video depth completion model $\mathcal{F}_{\text{dep}}$ introduced in the main paper. The model is initialized from Wan2.1-Fun-1.3B-InP and adopts a dual-path conditioning architecture consisting of a geometry-anchored denoising branch and an auxiliary RGB guidance branch.

Condition encoding. Following Eq. 5, the cleaned partial cache depth $\overline{\mathbf{D}}^i$ is converted into the normalized partial disparity $\overline{\mathbf{U}}^i$. We independently encode $\overline{\mathbf{U}}^i$ and the generated RGB video $\mathbf{V}^i$ using the pretrained Wan VAE encoder $\mathcal{E}_{\text{vae}}$:

$$\begin{aligned} \mathbf{z}^i_{\text{partial}} &= \mathcal{E}_{\text{vae}}\left(\text{Rep}_3\left(\overline{\mathbf{U}}^i\right)\right), \\ \mathbf{z}^i_{\text{rgb}} &= \mathcal{E}_{\text{vae}}\left(\mathbf{V}^i\right), \end{aligned} \tag{47}$$

where $\text{Rep}_3(\cdot)$ replicates the single-channel partial disparity over three channels before VAE encoding. Both $\mathbf{z}^i_{\text{partial}}$ and $\mathbf{z}^i_{\text{rgb}}$ contain 16 latent channels.

Geometry-anchored denoising branch. Let $\mathbf{z}_\tau^i$ denote the noisy disparity latent at diffusion timestep τ. Following the Wan inpainting interface, we concatenate it with the partial-disparity latent and a four-channel inpainting indicator:

$$\mathbf{h}_\tau^i = \mathrm{Concat}_c \left[\mathbf{z}_\tau^i, \mathbf{m}_{\mathrm{inp}}, \mathbf{z}_{\mathrm{partial}}^i\right], \tag{48}$$

where Concat_c denotes channel-wise concatenation and $\mathbf{m}_{\mathrm{inp}}$ is set to one over the complete latent volume. The resulting 36-channel tensor consists of 16 noisy disparity channels, four inpainting-indicator channels, and 16 partial-disparity channels.

The main-branch tokens are obtained through the original spatio-temporal patch embedding of the Wan inpainting model:

$$\mathbf{x}_0^i = \mathcal{E}_{\mathrm{patch}}^{\mathrm{dep}} \left(\mathbf{h}_\tau^i\right). \tag{49}$$

This branch directly incorporates the partial disparity into the main denoising stream, thereby anchoring the prediction to the geometry accumulated in the 4D memory.

Auxiliary RGB guidance branch. The generated RGB video provides complementary semantic, structural, and temporal cues for regions not covered by the partial cache disparity. Following our implementation, we replicate its 16-channel latent three times along the channel dimension and process the resulting 48-channel tensor using an additional spatio-temporal patch embedding:

$$\mathbf{g}_0^i = \mathcal{E}_{\mathrm{patch}}^{\mathrm{rgb}} \left(\mathrm{Concat}_c \left[\mathbf{z}_{\mathrm{rgb}}^i, \mathbf{z}_{\mathrm{rgb}}^i, \mathbf{z}_{\mathrm{rgb}}^i\right]\right). \tag{50}$$

The auxiliary branch contains six DiT blocks and progressively updates the RGB guidance features at every fifth layer of the main Wan backbone. Before each main DiT block, an RGB guidance feature is projected through a learned linear layer and added to the main hidden state. Let $\mathbf{x}_\ell^i$ denote the hidden state before the ℓ-th main DiT block, and let $\mathbf{r}_\ell^i$ denote the projected RGB guidance feature injected at this layer. The feature injection and backbone update are expressed as

$$\begin{aligned} \widehat{\mathbf{x}}_\ell^i &= \mathbf{x}_\ell^i + \mathbf{r}_\ell^i, \\ \mathbf{x}_{\ell+1}^i &= \mathcal{B}_\ell \left(\widehat{\mathbf{x}}_\ell^i, \mathbf{e}_{\mathrm{text}}, \tau\right), \end{aligned} \tag{51}$$

where $\widehat{\mathbf{x}}_\ell^i$ is the hidden state after RGB guidance injection, $\mathcal{B}_\ell$ denotes the ℓ-th main DiT block, and $\mathbf{e}_{\mathrm{text}}$ is the embedding of the input-video caption. At layers $\ell \in \{0, 5, 10, 15, 20, 25\}$, the RGB guidance features are first updated by the corresponding auxiliary DiT block; the remaining layers receive learned linear projections of the initial RGB guidance tokens. The main and auxiliary branches use the same diffusion timestep conditioning, text context, and three-dimensional rotary position embeddings.

Dense disparity decoding. After flow-matching denoising, the clean disparity latent $\mathbf{z}_0^i$ is decoded using the Wan VAE decoder $\mathcal{D}_{\mathrm{vae}}$. Since the decoder produces a three-channel output, we average its channels and apply min–max normalization to obtain the dense relative disparity:

$$\widehat{\mathbf{U}}^i = \mathcal{N} \left(\frac{1}{3} \sum_{c=1}^{3} \left[\mathcal{D}_{\mathrm{vae}} \left(\mathbf{z}_0^i\right)\right]_c\right). \tag{52}$$

The predicted disparity $\widehat{\mathbf{U}}^i$ is subsequently converted to metric depth through the two-stage global alignment described below, followed by Poisson refinement to obtain the final cache-consistent dense depth video $\mathbf{D}^i$.

Cache-supported region. Let $\overline{\mathbf{D}}^i$ denote the partial metric depth rendered from the 4D memory. We define the cache-supported region as

$$\Omega^i = \left\{(t, \mathbf{p}) \;\middle|\; \overline{\mathbf{M}}_t^i(\mathbf{p}) = 1, \quad \overline{\mathbf{D}}_t^i(\mathbf{p}) > \tau_{\mathrm{valid}}\right\}, \tag{53}$$

where $\tau_{\mathrm{valid}} = 10^{-6}$. Only pixels that are both visible in the cache rendering and associated with valid depth values are used as geometric anchors. No additional mask erosion is applied in our default configuration.

Global affine alignment. In the first stage, we normalize the valid values of the partial cache depth over the complete video:

$$\overline{\mathbf{D}}^i_{\mathrm{n}} = \mathcal{N}\left(\overline{\mathbf{D}}^i; \Omega^i\right), \tag{54}$$

where $\mathcal{N}(\cdot; \Omega^i)$ denotes min–max normalization over Ω^i. To reduce the influence of outliers, we retain only values between the 2% and 98% quantiles of $\overline{\mathbf{D}}^i_{\mathrm{n}}$. Denoting the corresponding inlier set by Ω^i_{in}, we estimate a global affine transformation shared across all frames:

$$\begin{aligned}\left(a^i_1, b^i_1\right) = \arg\min_{a,b} \quad & \sum_{(t,\mathbf{p})\in\Omega^i_{\mathrm{in}}} \left(a\widehat{\mathbf{U}}^i_t(\mathbf{p}) + b - \overline{\mathbf{D}}^i_{\mathrm{n},t}(\mathbf{p})\right)^2 \\ & + \lambda_{\mathrm{ridge}}\left(a^2 + b^2\right),\end{aligned} \tag{55}$$

where $\lambda_{\mathrm{ridge}} = 10^{-4}$. If fewer than 100 inlier pixels are available, we use the identity transformation. The aligned normalized prediction is

$$\mathbf{A}^i = \mathrm{clip}\left(a^i_1\widehat{\mathbf{U}}^i + b^i_1, 0, 1\right). \tag{56}$$

In the second stage, we recover the metric scale by fitting $\mathbf{A}^i$ to the original partial cache depth:

$$\left(a^i_2, c^i_2\right) = \arg\min_{a,c} \sum_{(t,\mathbf{p})\in\Omega^i} \left(a\mathbf{A}^i_t(\mathbf{p}) + c - \overline{\mathbf{D}}^i_t(\mathbf{p})\right)^2. \tag{57}$$

This regression uses all cache-supported pixels without quantile trimming or ridge regularization. The resulting dense metric-depth prediction is

$$\mathbf{G}^i = a^i_2\mathbf{A}^i + c^i_2. \tag{58}$$

Both affine transformations are shared across the complete video, which avoids frame-dependent scale fluctuations. We do not directly overwrite $\mathbf{G}^i$ in cache-supported regions; exact consistency is imposed by the subsequent Poisson refinement.

Poisson refinement. We solve the metric-depth Poisson objective in Eq. 7 independently for each frame. The solution is initialized using the partial cache depth in known regions and the aligned prediction elsewhere:

$$\mathbf{X}^{(0)}_t(\mathbf{p}) = \begin{cases}\overline{\mathbf{D}}^i_t(\mathbf{p}), & (t,\mathbf{p}) \in \Omega^i, \\ \mathbf{G}^i_t(\mathbf{p}), & (t,\mathbf{p}) \notin \Omega^i.\end{cases} \tag{59}$$

We discretize the Poisson equation using a four-connected neighborhood. For an unknown pixel $\mathbf{p} \notin \Omega^i_t$, one Jacobi iteration is

$$\mathbf{X}^{(k+1)}_t(\mathbf{p}) = \frac{1}{4}\left[\sum_{\mathbf{q}\in\mathcal{N}_4(\mathbf{p})} \mathbf{X}^{(k)}_t(\mathbf{q}) + 4\mathbf{G}^i_t(\mathbf{p}) - \sum_{\mathbf{q}\in\mathcal{N}_4(\mathbf{p})} \mathbf{G}^i_t(\mathbf{q})\right], \tag{60}$$

where $\mathcal{N}_4(\mathbf{p})$ denotes the four-connected neighborhood. After every iteration, cache-supported pixels are reset to their metric cache-depth values:

$$\mathbf{X}^{(k+1)}_t(\mathbf{p}) = \overline{\mathbf{D}}^i_t(\mathbf{p}), \qquad \forall\,(t,\mathbf{p}) \in \Omega^i. \tag{61}$$

We use replicated padding at image boundaries and perform 800 Jacobi iterations. For frames without cache-supported pixels, $\mathbf{G}^i_t$ is retained unchanged. The final dense metric-depth video is

$$\mathbf{D}^i = \left\{\mathbf{X}^{(800)}_t\right\}_{t=1}^{T}. \tag{62}$$

The two-stage alignment recovers the metric scale of the 4D memory, while the Poisson refinement preserves the completed depth structure and enforces exact consistency with cached geometry.

B.4 DUAL-STREAM WARPING

As described in the main paper, the appearance and geometry streams are two logical views of the same accumulated 4D memory rather than two physically duplicated point-cloud representations. Both streams share the RGB-D observations, camera poses, and provenance masks of the input video and previously generated trajectories. Their difference lies only in the source-selection masks used during point-cloud warping.

Stream-specific source masks. Let $\mathbf{M}_t^j$ denote the provenance mask stored for the j-th generated trajectory at time t, where visible cache-supported regions are marked as one. For the original input trajectory $j = 0$, all pixels are treated as valid in both streams. For historical generated trajectories $j > 0$, the source masks of the two streams are defined as

$$\begin{aligned} \mathbf{S}_{\mathrm{app},t}^0 &= \mathbf{S}_{\mathrm{geo},t}^0 = \mathbf{1}, \\ \mathbf{S}_{\mathrm{app},t}^j &= \mathbf{1} - \mathrm{Dilate}_{3\times 3}\left(\mathbf{M}_t^j\right), \\ \mathbf{S}_{\mathrm{geo},t}^j &= \mathbf{1}, \qquad j > 0. \end{aligned} \tag{63}$$

We apply one 3×3 dilation to the stored provenance mask before inversion. Consequently, the appearance stream retains the original input together with newly synthesized regions from previous trajectories, whereas the geometry stream uses all valid RGB-D observations.

Parallel point-cloud warping. Using the elementary warping operator $\mathcal{W}$ and cross-trajectory z-buffer composition introduced in Sec. A.2, the two streams are rendered independently:

$$\begin{aligned} \left(\mathbf{V}_{\mathrm{app}}^i, \mathbf{D}_{\mathrm{app}}^i, \mathbf{M}_{\mathrm{app}}^i\right) &= \mathrm{ZBuf}\left(\left\{\mathcal{W}\left(\mathbf{I}_t^j, \mathbf{D}_t^j, \mathbf{S}_{\mathrm{app},t}^j, \mathbf{C}_t^j, \mathbf{C}_t^i, \mathbf{K}\right)\right\}_{j=0}^{i-1}\right)_{t=1}^{T}, \\ \left(\mathbf{V}_{\mathrm{geo}}^i, \mathbf{D}_{\mathrm{geo}}^i, \mathbf{M}_{\mathrm{geo}}^i\right) &= \mathrm{ZBuf}\left(\left\{\mathcal{W}\left(\mathbf{I}_t^j, \mathbf{D}_t^j, \mathbf{S}_{\mathrm{geo},t}^j, \mathbf{C}_t^j, \mathbf{C}_t^i, \mathbf{K}\right)\right\}_{j=0}^{i-1}\right)_{t=1}^{T}. \end{aligned} \tag{64}$$

Both streams use the same depth-aware bilinear splatting and nearest-depth z-buffer. In our implementation, the observations are processed in chunks of 16 frames and eight source trajectories to reduce peak memory consumption without changing the rendering result.

Depth-conflict detection. The rendered metric depths are converted to the bounded inverse-depth representation

$$\phi(d) = \frac{1}{d+1}. \tag{65}$$

We identify conflicts by comparing the quantized inverse depths of the two streams:

$$\mathbf{Q}_t^i(\mathbf{p}) = \mathbb{I}\Big[\big|\,\mathrm{round}\left(255\,\phi(\mathbf{D}_{\mathrm{app},t}^i(\mathbf{p}))\right) - \mathrm{round}\left(255\,\phi(\mathbf{D}_{\mathrm{geo},t}^i(\mathbf{p}))\right)\big| > 15\Big]. \tag{66}$$

Thus, the discrepancy threshold corresponds approximately to $\tau_{\mathrm{occ}} = 15/255$ in the normalized inverse-depth domain.

Pixel-wise stream selection. At pixels without detected conflicts, we preserve the sharper appearance-stream result. At conflicting pixels, we select the geometry stream, whose denser point coverage provides more reliable visibility reasoning:

$$\begin{aligned} \widetilde{\mathbf{V}}^i &= \left(1 - \mathbf{Q}^i\right) \odot \mathbf{V}_{\mathrm{app}}^i + \mathbf{Q}^i \odot \mathbf{V}_{\mathrm{geo}}^i, \\ \widetilde{\mathbf{D}}^i &= \left(1 - \mathbf{Q}^i\right) \odot \mathbf{D}_{\mathrm{app}}^i + \mathbf{Q}^i \odot \mathbf{D}_{\mathrm{geo}}^i, \\ \widetilde{\mathbf{M}}^i &= \left(1 - \mathbf{Q}^i\right) \odot \mathbf{M}_{\mathrm{app}}^i + \mathbf{Q}^i \odot \mathbf{M}_{\mathrm{geo}}^i. \end{aligned} \tag{67}$$

The resulting $(\widetilde{\mathbf{V}}^i, \widetilde{\mathbf{D}}^i, \widetilde{\mathbf{M}}^i)$ correspond exactly to the output of $\mathcal{R}_{\mathrm{dual}}$ in the main paper.

B.5 GEOMETRY-AWARE VIDEO REFINEMENT

We provide the architectural details of the geometry-aware video refinement model $\mathcal{F}_{\text{ref}}$ introduced in the main paper. The model is initialized from Wan2.1-Fun-1.3B-InP and follows the same latent video diffusion formulation as $\mathcal{F}_{\text{app}}$. It complements the initial appearance generation stage by using the recovered dense geometry $\mathbf{D}^i$ to suppress residual texture flicker and high-frequency artifacts before the generated RGB-D observations are written into the 4D memory.

Geometry-aware condition encoding. The refinement model takes the cleaned partial RGB rendering $\overline{\mathbf{V}}^i$ and completed dense depth video $\mathbf{D}^i$ as its effective conditions. Both videos are encoded using the pretrained Wan VAE encoder:

$$\begin{aligned} \mathbf{z}^i_{\text{rgb,ref}} &= \mathcal{E}_{\text{vae}}\left(\overline{\mathbf{V}}^i\right), \\ \mathbf{z}^i_{\text{dep,ref}} &= \mathcal{E}_{\text{vae}}\left(\text{Rep}_3\left(\mathbf{D}^i\right)\right), \end{aligned} \tag{68}$$

where $\text{Rep}_3(\cdot)$ replicates the single-channel depth video over three channels before VAE encoding. The two conditions are thus represented in the same spatio-temporal latent space as the target RGB video.

Latent-space condition injection. Let $\mathbf{z}^i_{\tau,\text{ref}}$ denote the noisy RGB latent at diffusion timestep τ. Following the Wan inpainting interface, we concatenate it with the partial-RGB latent, dense-depth latent, and a constant conditioning latent:

$$\mathbf{h}^i_{\tau,\text{ref}} = \text{Concat}_c\left[\mathbf{z}^i_{\tau,\text{ref}}, \mathbf{z}^i_{\text{rgb,ref}}, \mathbf{z}^i_{\text{dep,ref}}, \mathbf{z}_{\text{const}}\right], \tag{69}$$

where $\mathbf{z}_{\text{const}}$ is obtained by encoding an all-one video and preserves the conditioning layout of the pretrained inpainting model without introducing additional scene information. Since each latent contains 16 channels, the resulting input contains 64 channels.

The concatenated latent is mapped to the Wan DiT hidden dimension through a 3D convolutional patch embedding:

$$\mathbf{x}^i_{\tau,\text{ref}} = \mathcal{E}^{\text{ref}}_{\text{patch}}\left(\mathbf{h}^i_{\tau,\text{ref}}\right). \tag{70}$$

The resulting tokens are processed by the Wan DiT blocks using full spatio-temporal self-attention, text cross-attention, diffusion timestep modulation, and three-dimensional rotary position embeddings. After flow-matching denoising, the clean RGB latent is decoded by the Wan VAE decoder, producing the refined video $\mathbf{V}^i_{\text{ref}}$.

Memory update. The partial RGB rendering preserves appearance already established in the 4D memory, while the completed depth provides dense structural guidance across both previously observed and newly revealed regions. Their combination allows $\mathcal{F}_{\text{ref}}$ to improve the appearance of $\mathbf{V}^i$ without modifying its recovered depth $\mathbf{D}^i$.

When refinement is enabled, the refined video replaces the initial appearance-generation result during memory update:

$$\mathbf{V}^i_{\text{upd}} = \begin{cases} \mathbf{V}^i_{\text{ref}}, & \text{with refinement,} \\ \mathbf{V}^i, & \text{without refinement.} \end{cases} \tag{71}$$

The RGB-D pair $(\mathbf{V}^i_{\text{upd}}, \mathbf{D}^i)$ is subsequently back-projected and fused into the 4D memory. Therefore, the refinement stage not only improves the current video but also prevents residual appearance artifacts from accumulating and propagating to subsequent camera trajectories.

C TRAINING AND INFERENCE DETAILS

C.1 TRAINING DATA CONSTRUCTION

We construct a unified training corpus containing 136K samples the from MultiCamVideo dataset and 100K samples from the OpenVid dataset. Following TrajectoryCrafter Yu et al. (2025b), we employ double reprojection to synthesize target-aligned partial RGB-D conditions from monocular videos. We use DepthCrafter Hu et al. (2025) to estimate temporally consistent depth sequences for the selected training clips.

Double reprojection. Let $\mathbf{V}_{\mathrm{gt}} = \{\mathbf{I}_{\mathrm{gt},t}\}_{t=1}^{T}$ denote a clean target video, $\mathbf{D}_{\mathrm{gt}} = \{\mathbf{D}_{\mathrm{gt},t}\}_{t=1}^{T}$ its DepthCrafter depth sequence, and $\mathbf{C}_{\mathrm{gt}} = \{\mathbf{C}_{\mathrm{gt},t}\}_{t=1}^{T}$ its camera trajectory. For every timestamp, we first construct the colored point cloud

$$\mathcal{P}_{\mathrm{gt},t} = \mathrm{BackProj}\left(\mathbf{I}_{\mathrm{gt},t}, \mathbf{D}_{\mathrm{gt},t}, \mathbf{C}_{\mathrm{gt},t}; \mathbf{1}\right), \tag{72}$$

where $\mathrm{BackProj}$ is the RGB-D back-projection operator introduced in the main paper.

We then sample a virtual camera trajectory $\mathbf{C}' = \{\mathbf{C}'_t\}_{t=1}^{T}$ and render the point-cloud sequence along this trajectory:

$$(\mathbf{I}'_t, \mathbf{D}'_t, \mathbf{M}'_t) = \mathcal{R}\left(\mathcal{P}_{\mathrm{gt},t}, \mathbf{C}'_t\right), \qquad t = 1, \ldots, T, \tag{73}$$

where $\mathcal{R}$ is the point-cloud rendering operator defined in Sec. A.2. This first reprojection introduces view-dependent disocclusions, out-of-view content, and sparse rendering holes.

We subsequently reconstruct a point cloud from each intermediate RGB-D frame:

$$\mathcal{P}'_t = \mathrm{BackProj}\left(\mathbf{I}'_t, \mathbf{D}'_t, \mathbf{C}'_t; \mathbf{M}'_t\right), \tag{74}$$

and render it back to the original target camera:

$$(\mathbf{I}_{\mathrm{par},t}, \mathbf{D}_{\mathrm{par},t}, \mathbf{M}_{\mathrm{par},t}) = \mathcal{R}\left(\mathcal{P}'_t, \mathbf{C}_{\mathrm{gt},t}\right). \tag{75}$$

Collecting the resulting frames gives

$$\begin{aligned} \mathbf{V}_{\mathrm{par}} &= \{\mathbf{I}_{\mathrm{par},t}\}_{t=1}^{T}, \\ \mathbf{D}_{\mathrm{par}} &= \{\mathbf{D}_{\mathrm{par},t}\}_{t=1}^{T}, \\ \mathbf{M}_{\mathrm{par}} &= \{\mathbf{M}_{\mathrm{par},t}\}_{t=1}^{T}. \end{aligned} \tag{76}$$

Although the second reprojection returns to the original camera trajectory, points removed by occlusion or field-of-view changes during the first reprojection cannot be recovered. Consequently, $\mathbf{V}_{\mathrm{par}}$ and $\mathbf{D}_{\mathrm{par}}$ contain realistic geometry-induced missing regions while remaining spatially aligned with $\mathbf{V}_{\mathrm{gt}}$ and $\mathbf{D}_{\mathrm{gt}}$.

Appearance video diffusion. For training the cache-guided appearance model $\mathcal{F}_{\mathrm{app}}$, we use the double-reprojected RGB-D conditions and visibility mask as inputs, with the clean video as supervision:

$$\mathcal{F}_{\mathrm{app}}\left(\mathbf{V}_{\mathrm{par}}, \mathbf{D}_{\mathrm{par}}, \mathbf{M}_{\mathrm{par}}\right) \longrightarrow \mathbf{V}_{\mathrm{gt}}. \tag{77}$$

Here, $(\mathbf{V}_{\mathrm{par}}, \mathbf{D}_{\mathrm{par}}, \mathbf{M}_{\mathrm{par}})$ plays the same role as $(\overline{\mathbf{V}}^{i}, \overline{\mathbf{D}}^{i}, \overline{\mathbf{M}}^{i})$ during inference.

Video depth completion. To ensure exact consistency between the partial-depth input and its supervision in known regions, we construct the partial depth by masking the complete DepthCrafter depth using the double-reprojection visibility mask:

$$\mathbf{D}_{\mathrm{mask}} = \mathbf{M}_{\mathrm{par}} \odot \mathbf{D}_{\mathrm{gt}}. \tag{78}$$

We then convert the masked and complete depths into normalized disparities:

$$\begin{aligned} \mathbf{U}_{\mathrm{par}} &= \mathcal{N}\left(\frac{1}{\mathbf{D}_{\mathrm{mask}} + \epsilon}; \mathbf{M}_{\mathrm{par}}\right), \\ \mathbf{U}_{\mathrm{gt}} &= \mathcal{N}\left(\frac{1}{\mathbf{D}_{\mathrm{gt}} + \epsilon}; \mathbf{1}\right), \end{aligned} \tag{79}$$

where $\mathcal{N}(\cdot; \mathbf{M})$ denotes min–max normalization over pixels selected by $\mathbf{M}$. The depth-completion model is trained using

$$\mathcal{F}_{\mathrm{dep}}\left(\mathbf{V}_{\mathrm{gt}}, \mathbf{U}_{\mathrm{par}}\right) \longrightarrow \mathbf{U}_{\mathrm{gt}}. \tag{80}$$

During training, $\mathbf{V}_{\mathrm{gt}}$ plays the role of the generated RGB video $\mathbf{V}^{i}$ in Eq. 6, while $\mathbf{U}_{\mathrm{par}}$ corresponds to the partial disparity $\overline{\mathbf{U}}^{i}$.

Geometry-aware refinement. For training the refinement model, we use the partial RGB video and complete DepthCrafter depth as conditions, with the clean target video as supervision:

$$\mathcal{F}_{\mathrm{ref}}\left(\mathbf{V}_{\mathrm{par}}, \mathbf{D}_{\mathrm{gt}}\right) \longrightarrow \mathbf{V}_{\mathrm{gt}}. \tag{81}$$

Here, $\mathbf{V}_{\mathrm{par}}$ corresponds to the partial RGB condition $\overline{\mathbf{V}}^i$, while $\mathbf{D}_{\mathrm{gt}}$ plays the role of the completed dense depth $\mathbf{D}^i$ used during inference. This construction teaches the refinement model to restore high-frequency appearance details while remaining constrained by complete scene geometry.

C.2 Training Configurations

We train the cache-guided appearance generation model $\mathcal{F}_{\mathrm{app}}$, cache-consistent video depth completion model $\mathcal{F}_{\mathrm{dep}}$, and geometry-aware refinement model $\mathcal{F}_{\mathrm{ref}}$ independently. All three models are initialized from the pretrained Wan2.1-Fun-1.3B-InP checkpoint and adopt the same training configuration.

For each model, we optimize all parameters of the video DiT while keeping the Wan VAE, UMT5 text encoder, and CLIP image encoder frozen. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 1×10^{-4}, and a weight decay of 1×10^{-2}. The learning rate follows a cosine annealing schedule without warm-up and decreases to 1% of its initial value, i.e., 1×10^{-6}, over the complete training process.

All models are trained for five epochs using bfloat16 precision. Each training sample contains 81 frames at a resolution of 480×832. We use a per-GPU batch size of one on eight NVIDIA H20 GPUs, resulting in a global batch size of eight without additional gradient accumulation. Each training sample is used once per epoch. We enable distributed data parallelism with unused-parameter detection and save the trainable video-DiT parameters every 1,000 optimization steps. The module-specific inputs and supervision targets follow the training-data construction described in Sec. C.

C.3 Inference Configurations

At inference time, Demiurge sequentially processes the prescribed target camera trajectories. For each trajectory, we first render the current 4D memory using dual-stream warping and remove unreliable pixels using the noise monitor. The cleaned partial RGB, depth, and visibility conditions are then passed to $\mathcal{F}_{\mathrm{app}}$ to generate the initial novel-view RGB video $\mathbf{V}^i$. Subsequently, $\mathcal{F}_{\mathrm{dep}}$ predicts a dense relative disparity, which is converted to metric depth through the two-stage global alignment and Poisson refinement. Finally, $\mathcal{F}_{\mathrm{ref}}$ uses the partial RGB rendering and completed dense depth to produce the refined video $\mathbf{V}^i_{\mathrm{ref}}$. The resulting RGB-D pair $(\mathbf{V}^i_{\mathrm{ref}}, \mathbf{D}^i)$ is back-projected and accumulated into the 4D memory before processing the next target trajectory.

All three video diffusion modules generate 81-frame videos at a resolution of 480×832. We use 50 flow-matching inference steps. All diffusion modules operate in bfloat16 precision. We enable tiled Wan VAE encoding and decoding with a tile size of $(30, 52)$ and a tile stride of $(15, 26)$ to reduce peak memory consumption.

To control peak memory during multi-source rendering, we process at most 16 consecutive frames and eight historical trajectories in each rendering batch. This chunking only changes the execution order and does not affect the z-buffer composition result. Rendered metric depth is encoded as bounded inverse depth using $\phi(d) = 1/(d+1)$ before being used as a diffusion condition. The geometric depth is recovered when the observation is subsequently used for point-cloud construction or warping.

D Evaluation Details

D.1 Metric Definitions

All quantitative metrics are computed over the complete evaluation set unless otherwise specified. The benchmark contains 128 input videos and 14 target camera trajectories per input, resulting in 1,792 generated novel-view videos.

Fréchet Video Distance. We use Fréchet Video Distance (FVD) to evaluate the distribution-level quality of generated videos. For every target trajectory, we compare the 128 generated videos with the 128 input videos. We use the pretrained I3D network as the feature extractor. All 81 frames of each video are resized to 224×224, normalized to $[-1, 1]$, and jointly passed to I3D to obtain a video-level feature vector.

Let $(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ and $(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ denote the mean and covariance of the generated and reference video features, respectively. FVD is computed as

$$\mathrm{FVD} = \|\boldsymbol{\mu}_g - \boldsymbol{\mu}_r\|_2^2 + \mathrm{Tr}\left(\boldsymbol{\Sigma}_g + \boldsymbol{\Sigma}_r - 2\left(\boldsymbol{\Sigma}_g\boldsymbol{\Sigma}_r\right)^{1/2}\right). \tag{82}$$

We compute FVD independently for each target trajectory and report the average over all trajectories.

Q-Align. We use Q-Align Wu et al. (2023) to assess the perceptual quality and aesthetics of the generated results at both the video and image levels. Specifically, we report four scores: Q-Align-Video Quality, Q-Align-Video Aesthetics, Q-Align-Image Quality, and Q-Align-Image Aesthetics.

For video-level evaluation, each 81-frame video is divided into consecutive segments of eight frames to satisfy the model context limit. Q-Align evaluates each segment independently, and the segment scores are averaged to obtain the video-level score. If the final segment contains only one frame, its last frame is repeated to form a two-frame segment. For image-level evaluation, all 81 frames are independently scored and then averaged. The reported values are further averaged over all scenes and target trajectories.

Projection MSE. Projection MSE (P-MSE) evaluates the consistency between a generated novel-view video and the content observable from the input video under the same target camera trajectory. For each input video and target trajectory, we first geometrically project the input RGB-D video onto the target trajectory, obtaining a partial RGB rendering and its visibility mask.

Let $\mathbf{Y}_{s,n,t}$ denote frame t of the video generated for scene s and target trajectory n, $\mathbf{R}_{s,n,t}$ the corresponding partial RGB projection of the input video, and $\mathbf{M}_{s,n,t}$ its binary visibility mask. All RGB values are normalized to $[0, 1]$. The P-MSE of an individual input–trajectory pair is

$$\text{P-MSE}_{s,n} = \frac{\displaystyle\sum_t \sum_{\mathbf{p}} \mathbf{M}_{s,n,t}(\mathbf{p}) \, \|\mathbf{Y}_{s,n,t}(\mathbf{p}) - \mathbf{R}_{s,n,t}(\mathbf{p})\|_2^2}{\displaystyle 3 \sum_t \sum_{\mathbf{p}} \mathbf{M}_{s,n,t}(\mathbf{p})}. \tag{83}$$

The factor of three averages the squared error over the RGB channels. The visibility mask is binarized at a grayscale threshold of $127/255$, and P-MSE is evaluated only over its visible regions.

We use pair-macro aggregation as the main metric, assigning equal weight to every input–trajectory pair:

$$\text{P-MSE} = \frac{1}{SN} \sum_{s=1}^{S} \sum_{n=1}^{N} \text{P-MSE}_{s,n}, \tag{84}$$

where $S = 128$ and $N = 14$. This differs from pooling all visible pixels globally, which would assign larger weights to pairs with greater projected coverage.

Camera trajectory estimation. We use VGGT Wang et al. (2025) to recover camera poses from the generated videos. For each scene and sampled timestamp, the frames generated along all 14 target trajectories are jointly processed to estimate their camera-to-world matrices. We sample every tenth frame from each 81-frame video. The estimated poses are compared against the target camera trajectories used during generation.

The target trajectories progressively interpolate from the source camera to their prescribed endpoints. Consequently, all target views share the same camera pose at the first frame. Since this timestamp contains no cross-view camera variation, we exclude it from camera accuracy evaluation.

Rotation error. Let $\widehat{\mathbf{R}}_{t,n}$ and $\mathbf{R}_{t,n}$ denote the estimated and target camera rotations at sampled timestamp t and trajectory n, respectively. VGGT and our trajectory generator use different camera-axis conventions and independent world-coordinate gauges. We therefore first determine a global

axis-convention rotation $\mathbf{D}$ from the 24 orientation-preserving signed axis permutations. For every sampled timestamp, we additionally estimate a world-frame alignment $\mathbf{W}_t$ through orthogonal Procrustes alignment.

The rotation error is defined as the mean geodesic distance:

$$\text{R-err} = \frac{1}{|\mathcal{T}|N} \sum_{t\in\mathcal{T}} \sum_{n=1}^{N} \arccos\left(\frac{\text{Tr}\left[\widehat{\mathbf{R}}_{t,n} \left(\mathbf{W}_t \mathbf{R}_{t,n} \mathbf{D}\right)^{\mathsf{T}}\right] - 1}{2} \right), \tag{85}$$

where $\mathcal{T}$ is the set of non-degenerate sampled timestamps. We report R-err in radians. The global convention alignment $\mathbf{D}$ is selected by exhaustive search to minimize the mean rotation error, while each $\mathbf{W}_t$ is solved in closed form.

Translation error. Let $\widehat{\mathbf{c}}_{t,n}$ and $\mathbf{c}_{t,n}$ denote the camera centers extracted from the estimated and target camera-to-world matrices, respectively. Since monocular camera reconstruction is ambiguous up to scale, rotation, and translation, we independently align the estimated camera centers to the target centers at each timestamp using Umeyama Sim(3) alignment:

$$\left(s_t, \mathbf{R}_t^{\text{sim}}, \mathbf{b}_t\right) = \arg\min_{s,\mathbf{R},\mathbf{b}} \sum_{n=1}^{N} \left\| \mathbf{c}_{t,n} - \left(s\mathbf{R}\widehat{\mathbf{c}}_{t,n} + \mathbf{b}\right) \right\|_2^2 . \tag{86}$$

Following the implementation, translation error is computed by summing the squared errors over trajectories and averaging over timestamps:

$$\text{T-err} = \frac{1}{|\mathcal{T}|} \sum_{t\in\mathcal{T}} \sum_{n=1}^{N} \left\| \mathbf{c}_{t,n} - \left(s_t \mathbf{R}_t^{\text{sim}} \widehat{\mathbf{c}}_{t,n} + \mathbf{b}_t\right) \right\|_2^2 . \tag{87}$$

Both R-err and T-err are first computed per scene and then averaged over all 128 evaluation scenes.

View Sync. View Sync evaluates the visual consistency among the novel-view videos generated from the same input sequence. For each scene, we select the temporally aligned final frame from each of the 14 generated videos and compute dense pairwise correspondences using RoMa Edstedt et al. (2024). This results in $\binom{14}{2} = 91$ view pairs per scene.

Before matching, all frames are resized to 480×832. For baselines that concatenate the generated result with an auxiliary rendering, we retain only the generated region before resizing. RoMa uses a coarse resolution of 560 and produces dense correspondences on an upsampled grid of 864×1152.

Let $\gamma_{s,n,m}(\mathbf{p})$ denote the confidence assigned by RoMa to the correspondence at pixel $\mathbf{p}$ between trajectories n and m of scene s. We count the high-confidence correspondences as

$$\text{VS}_{s,n,m} = \sum_{\mathbf{p}\in\Omega_{\text{RoMa}}} \mathbb{I}\left[\gamma_{s,n,m}(\mathbf{p}) > \tau_{\text{sync}}\right], \tag{88}$$

where Ω_{RoMa} denotes the dense output grid and $\tau_{\text{sync}} = 0.7$. Since RoMa performs symmetric matching, we retain only one correspondence direction to avoid duplicate counting.

The final View Sync score is the average number of high-confidence correspondences over all scenes and trajectory pairs:

$$\text{ViewSync} = \frac{1}{S\binom{N}{2}} \sum_{s=1}^{S} \sum_{1\le n<m\le N} \text{VS}_{s,n,m}, \tag{89}$$

where $S = 128$ and $N = 14$. A higher View Sync score indicates stronger cross-view consistency among videos generated along different camera trajectories. The optional fundamental-matrix verification is used only for diagnostic analysis and is not included in the reported View Sync score.

3DGS Reconstruction Consistency. We evaluate whether the multi-trajectory videos generated from the same input can be jointly explained by a coherent 3D representation. Specifically, we map all generated videos onto a common 81-frame reference timeline and select three normalized time points, $\mathcal{T} = \{1/3, 2/3, 1\}$. At each time point, we extract the temporally aligned frames from the 14

generated videos and independently reconstruct a 3D Gaussian representation using AnySplat Jiang et al. (2025). Different time points are reconstructed separately to avoid mixing dynamic scene content into a single static representation.

The reconstructed Gaussians are rendered back to the 14 input viewpoints using the camera poses and intrinsics estimated by AnySplat. We then compare each rendered image with its corresponding generated frame using PSNR, SSIM, and LPIPS. For a scene s and time point p, the reconstruction score is

$$\mathcal{E}^m_{s,p} = \frac{1}{N}\sum_{i=1}^{N} m\left(\hat{\mathbf{I}}^i_{s,p}, \mathbf{I}^i_{s,p}\right), \tag{90}$$

where $N = 14$, $\mathbf{I}^i_{s,p}$ is the frame extracted from the i-th generated trajectory, $\hat{\mathbf{I}}^i_{s,p}$ is its 3DGS rendering, and m denotes PSNR, SSIM, or LPIPS. The final result is obtained by first averaging over the three time points within each scene and then averaging equally over all 128 scenes:

$$\mathcal{E}^m = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{|\mathcal{T}|}\sum_{p\in\mathcal{T}}\mathcal{E}^m_{s,p}, \tag{91}$$

where $S = 128$. Higher PSNR and SSIM and lower LPIPS indicate that the generated views can be more faithfully represented by a shared 3D structure, reflecting stronger cross-view appearance and geometric consistency.

E KEY CAPABILITIES

We next qualitatively examine three key capabilities of Demiurge that underpin navigable 4D world generation: coherent spatiotemporal lattice construction, large-parallax video rerendering, and geometry-faithful 3D/4D reconstruction.

E.1 COHERENT SPATIOTEMPORAL LATTICE GENERATION

Figs. 5 and 6 illustrate the spatiotemporal lattices generated along ten camera trajectories. Each row corresponds to a target trajectory, while the columns show temporally aligned frames, jointly spanning the view and time dimensions. Demiurge preserves scene appearance, dynamic content, and recovered geometry across this lattice, demonstrating coherent world expansion beyond independently generated camera paths.

E.2 LARGE-PARALLAX VIDEO RERENDERING

As shown in Fig. 7, Demiurge supports progressive camera rotations of up to 180° in both directions. Despite the diminishing overlap with the input view, it consistently expands newly revealed regions while preserving scene identity, dynamics, and spatial continuity across successive trajectories.

E.3 GEOMETRY-FAITHFUL OUTPUTS FOR 3DGS RECONSTRUCTION

As shown in Figs. 8 and 9, we independently reconstruct a 3DGS from the generated multi-view observations at three time steps using AnySplat Jiang et al. (2025). The resulting representations preserve coherent scene geometry and produce consistent RGB and depth renderings across views, demonstrating that Demiurge generates geometrically compatible observations rather than merely plausible individual videos.

F MORE QUALITATIVE COMPARISONS

We provide additional qualitative comparisons with all competing methods in Figs. 10–15. In each figure, rows correspond to different target camera trajectories, while sampled frames are arranged chronologically from left to right. These examples cover diverse indoor and outdoor scenes, dynamic foreground objects, and substantial viewpoint changes.

Although the competing methods generally produce plausible videos along an individual trajectory, their independently generated views often exhibit appearance drift, object deformation, inconsistent background structure, or insufficient response to the prescribed camera motion. Such discrepancies become increasingly evident when comparing the same object or region across different trajectories. In contrast, Demiurge consistently preserves object identity, fine-grained appearance, scene layout, and temporal dynamics while revealing geometrically plausible content under novel viewpoints. These results further demonstrate the benefit of explicitly consolidating and reusing historical appearance and geometry through the evolving 4D memory.

G Additional ablation results

We provide additional qualitative ablations in Figs. 16–19. In each example, sampled frames are arranged chronologically from left to right, with the corresponding depth maps shown below.

Without the noise monitor, unreliable point-cloud projections near depth discontinuities propagate into the generated RGB video and recovered geometry, causing local boundary artifacts. Filtering these projections produces cleaner appearance and more stable object geometry, as shown in Fig. 16. Fig. 17 further demonstrates the importance of cache-consistent video depth completion. Direct depth estimation yields temporally unstable shapes and geometric discontinuities, whereas our module anchors observed regions to the cached geometry and consistently completes newly revealed regions. As shown in Fig. 18, geometry-aware refinement suppresses residual appearance artifacts and texture flicker while preserving the recovered scene structure. Finally, Fig. 19 verifies the complementary roles of dual-stream warping. Removing the geometry stream weakens occlusion reasoning and introduces erroneous projections, while removing the appearance stream causes visual degradation from repeatedly updated content. Their combination preserves stable appearance while retaining dense geometry for reliable visibility reasoning.

H Compatibility with Existing Method

To examine the modularity of Demiurge, we replace our cache-guided appearance video diffusion model with the existing GEN3C generator while retaining the remaining depth recovery and memory-update pipeline. As shown in Figs. 20–23, the resulting system generates coherent RGB videos across multiple trajectories and recovers temporally stable depth for subsequent memory updates. This demonstrates that our framework is not tied to a specific appearance generator: the explicit 4D memory and generate–recover–update cycle can serve as a general consistency layer for existing methods. Consequently, improvements in the underlying appearance generator can be incorporated to further enhance visual quality without redesigning the overall world-building pipeline.

I Additional visualizations

We present additional results on diverse indoor and outdoor dynamic scenes in Figs. 24–32. In each figure, rows correspond to different target camera trajectories and columns show temporally aligned samples arranged from left to right. The generated RGB frames and their recovered depths are shown in the upper and lower rows, respectively. Across different viewpoints and time steps, Demiurge consistently preserves scene appearance, dynamic content, and geometric structure, further demonstrating its ability to construct coherent multi-trajectory observations for dynamic scenes.

J Runtime Analysis

Under our default inference setting, the appearance video diffusion, video depth completion, and optional geometry-aware refinement modules each require approximately 1.5 minutes to process an 81-frame video at a resolution of 480×832. With refinement enabled, generating one target trajectory takes approximately 5 minutes end to end. Since target trajectories are processed autoregressively, the total inference time scales approximately linearly with their number, i.e., about $5N$ minutes for N trajectories. Refinement can be disabled when faster inference is preferred, reducing the diffusion inference time to approximately 3 minutes per trajectory. Although the current implementation is not optimized for speed, Demiurge remains stable over substantially longer generation

horizons than those used in our benchmark. In particular, we have generated up to 100 consecutive camera trajectories without noticeable error accumulation, demonstrating the long-horizon scalability of the evolving 4D memory.

K LIMITATIONS AND FUTURE WORK

The primary limitation of Demiurge is its inference latency. The current pipeline comprises several sequentially coupled modules, including appearance generation, video depth completion, optional geometry-aware refinement, and 4D memory update. Their intermediate RGB, depth, and geometric representations must be exchanged and synchronized before the next autoregressive step, preventing straightforward parallel execution. Consequently, the full pipeline requires approximately five minutes to generate one target-trajectory video, which limits its applicability to interactive scenarios.

An important direction for future work is to decouple latency-sensitive RGB generation from background geometry recovery and memory maintenance through an asynchronous cloud–edge architecture. Since users only consume the final RGB video, a fast path could immediately query the latest 4D cache and relevant video memories to synthesize the requested view, while a slower background path performs depth completion, optional refinement, back-projection, and memory updates asynchronously. The fast path would support responsive rendering within the currently cached region, whereas the background path would continuously consolidate newly generated observations and expand the renderable world. Such a fast–slow design could substantially reduce perceived latency without discarding the geometric consistency provided by the complete generate–recover–update cycle.

Moreover, developing a more efficient representation of the evolving 4D memory is equally important. Our current time indexed point-cloud representation prioritizes explicitness and geometric interpretability, but repeated observations may introduce redundant points and increase storage, retrieval, and rendering costs as the number of trajectories grows. Future work could jointly align and compress historical observations into a hierarchical spatiotemporal representation, merging geometrically and temporally compatible content while preserving dynamic boundaries and newly revealed regions. Confidence-aware fusion, adaptive spatial discretization, and compact 4D primitives may further reduce memory overhead and accelerate memory queries, enabling longer and larger-scale world expansion.



Figure 5: **Coherent spatiotemporal lattice generation (Case 1).** Rows correspond to different camera trajectories and columns to temporally aligned frames; each sample shows the generated RGB view and its recovered depth.

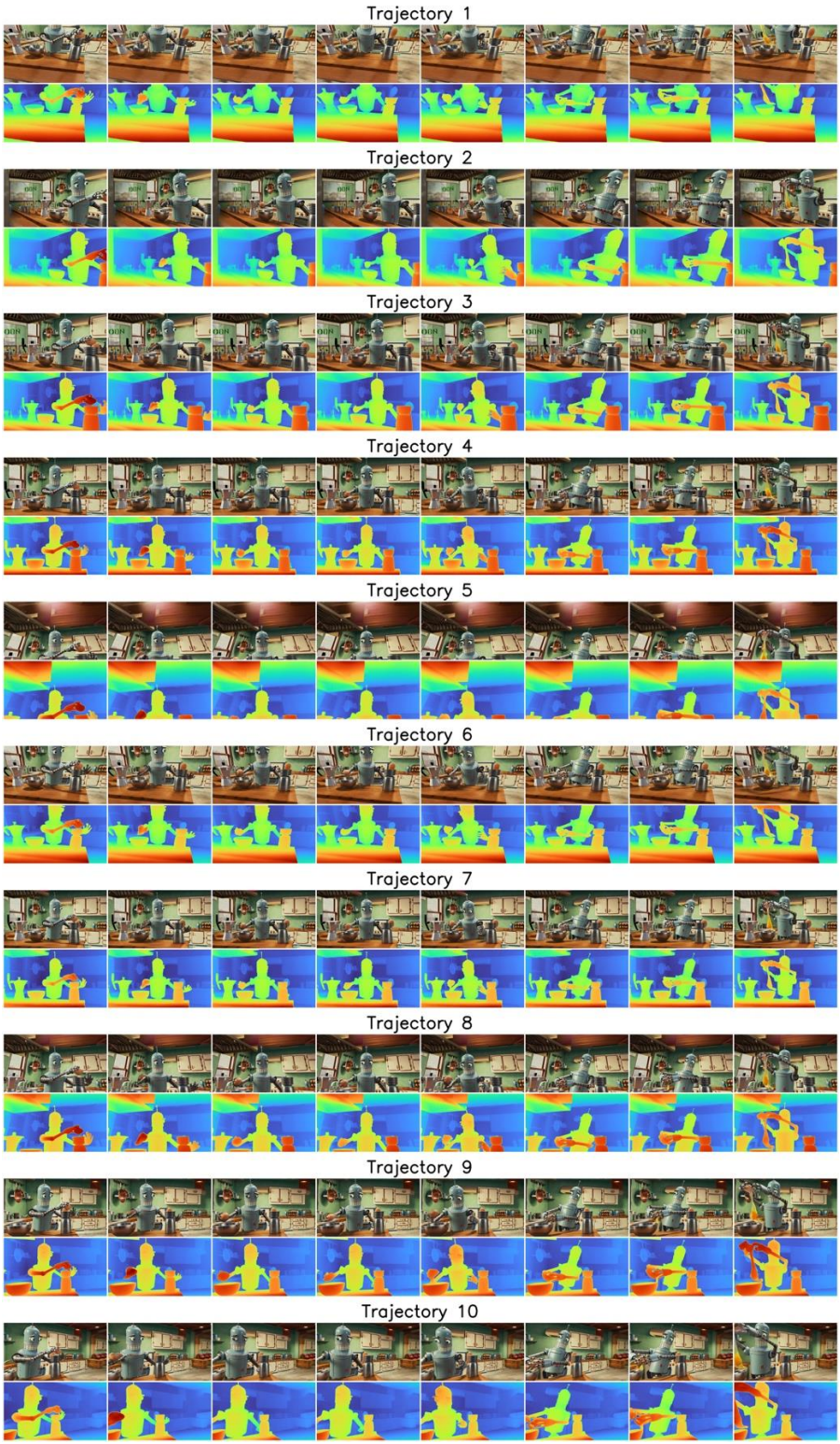
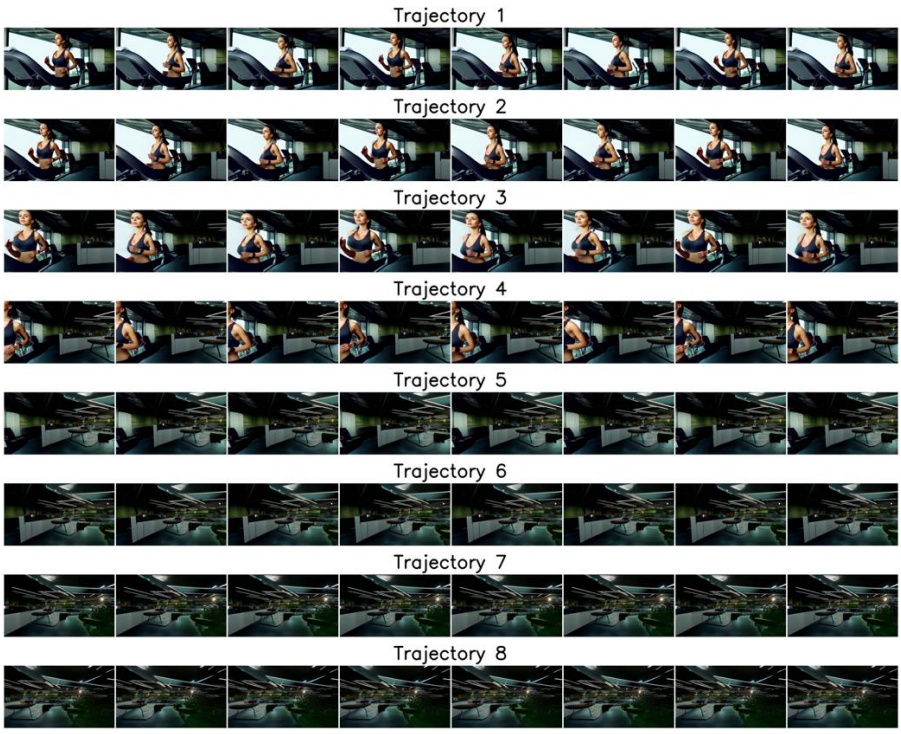
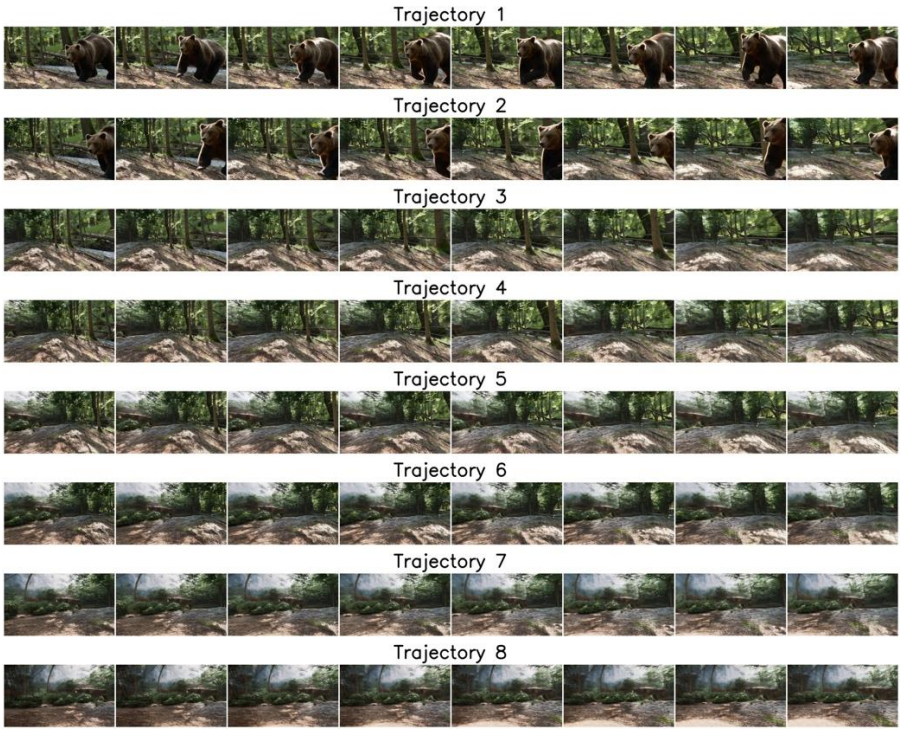


Figure 6: **Coherent spatiotemporal lattice generation (Case 2).** Rows correspond to different camera trajectories and columns to temporally aligned frames; each sample shows the generated RGB view and its recovered depth.

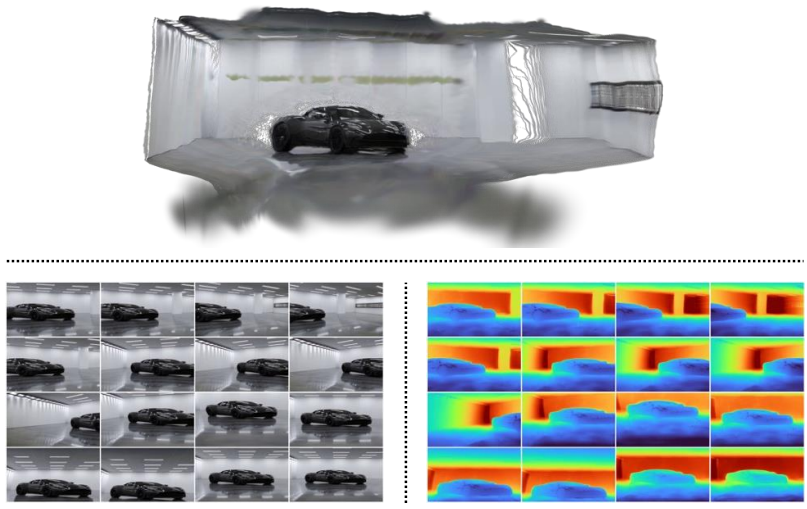


(a) Large parallax video rerendering of case 1.

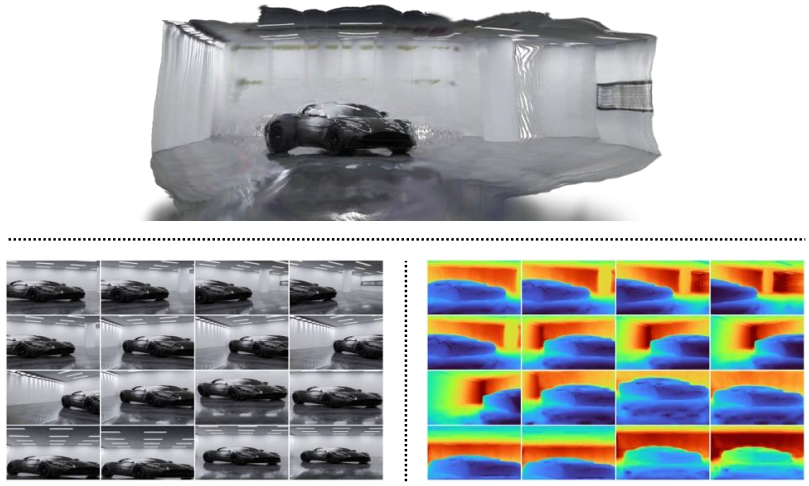


(b) Large parallax video rerendering of case 2.

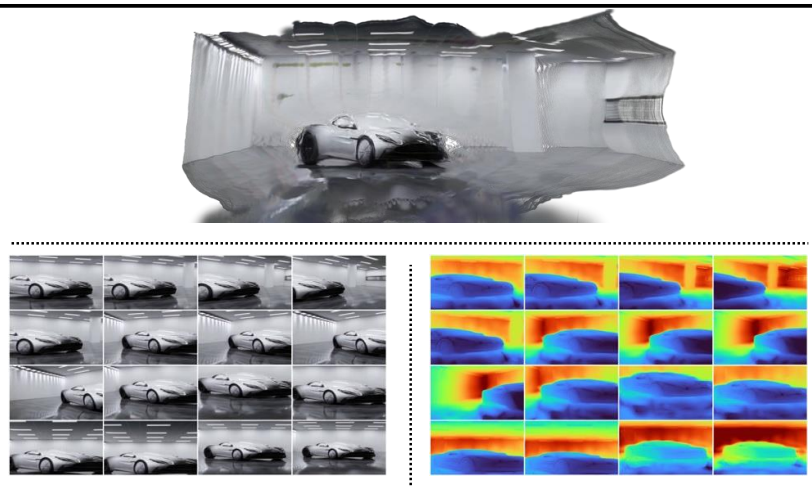
Figure 7: **Large-parallax video rerendering.** Demiurge progressively rotates the camera by up to 180° rightward **(a)** and leftward **(b)**, while maintaining consistent scene content across eight successive trajectories. Sampled frames in each row are ordered chronologically.



(a) 3DGS reconstruction and RGB–depth renderings at **timestep 1**.

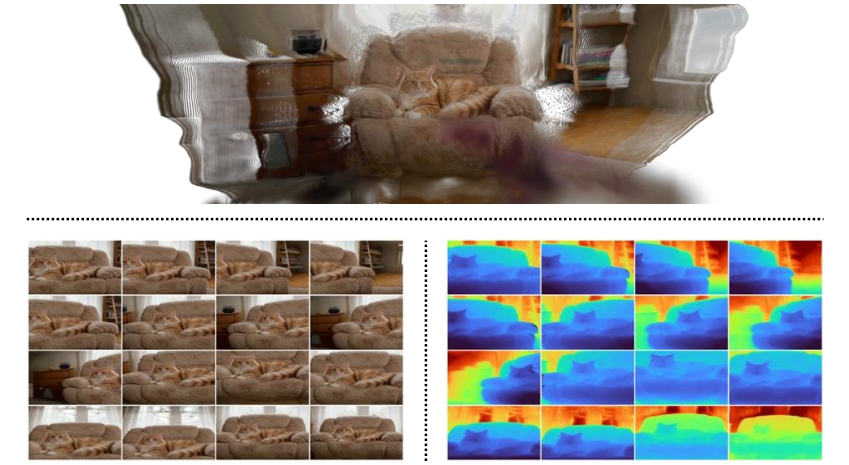


(b) 3DGS reconstruction and RGB–depth renderings at **timestep 2**.

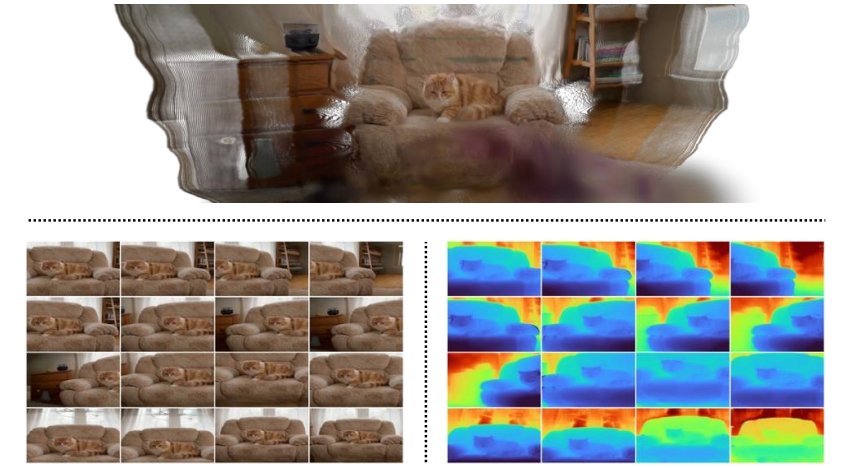


(c) 3DGS reconstruction and RGB–depth renderings at **timestep 3**.

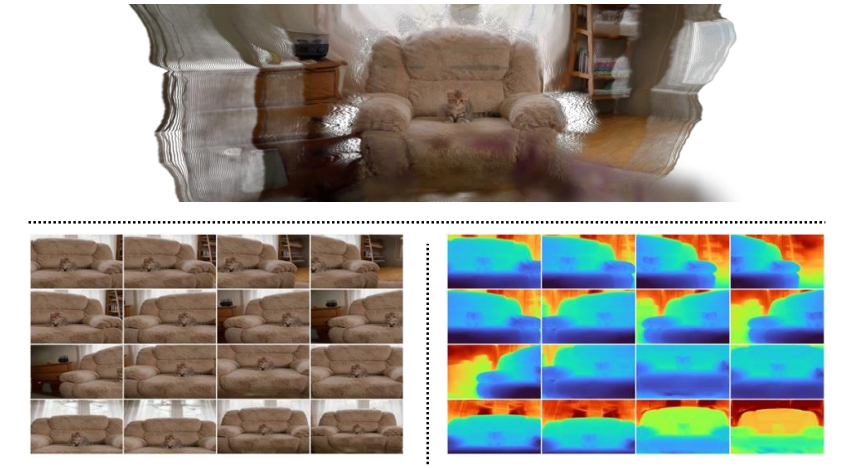
Figure 8: **Geometry-faithful 3DGS reconstruction (Case 1).** At each time step, we show the reconstructed 3DGS (top) and its multi-view RGB (bottom left) and depth renderings (bottom right).



(a) 3DGS reconstruction and RGB–depth renderings at **timestep 1**.

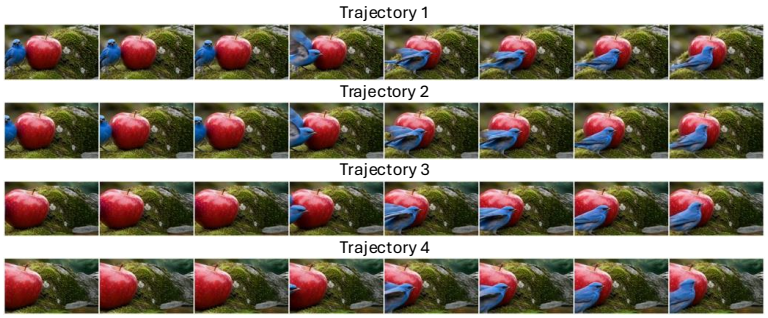


(b) 3DGS reconstruction and RGB–depth renderings at **timestep 2**.

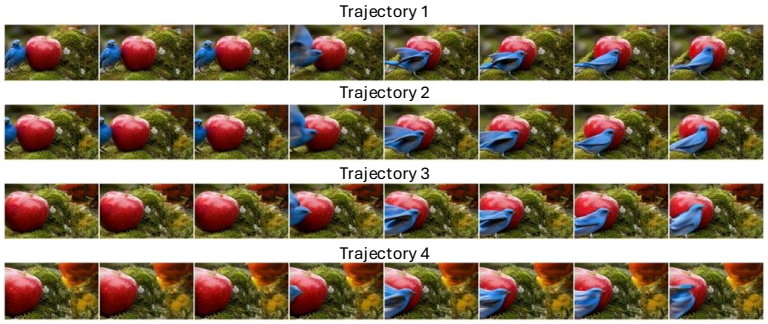


(c) 3DGS reconstruction and RGB–depth renderings at **timestep 3**.

Figure 9: **Geometry-faithful 3DGS reconstruction (Case 2).** At each time step, we show the reconstructed 3DGS (top) and its multi-view RGB (bottom left) and depth renderings (bottom right).

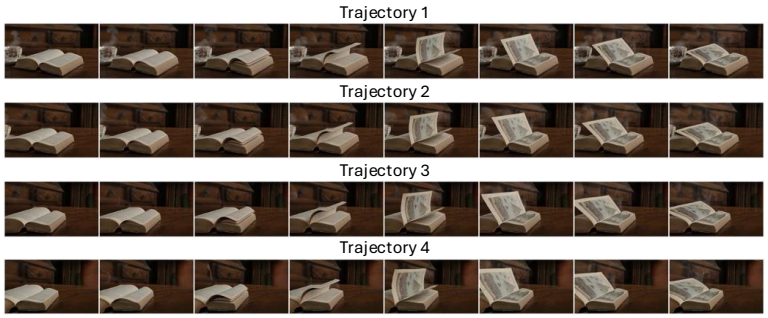


(a) Novel-view videos generated by **ours**.

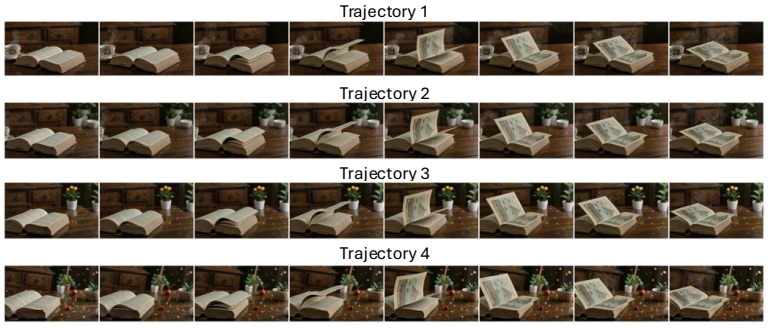


(b) Novel-view videos generated by **TrajCrafter**.

Figure 10: **Additional comparison with TrajCrafter.** Demiurge better preserves object appearance and scene geometry across different camera trajectories, with less content drift and viewpoint-dependent distortion.

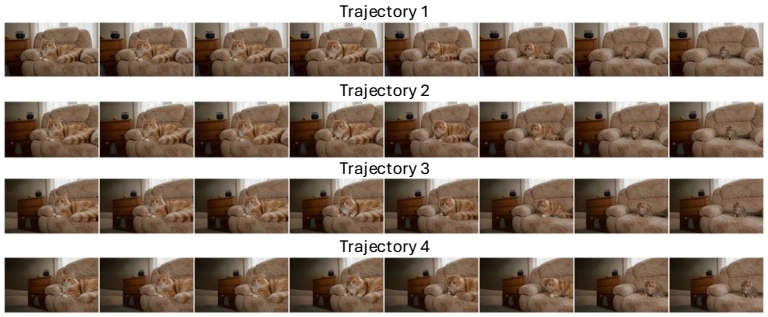


(a) Novel-view videos generated by **ours**.

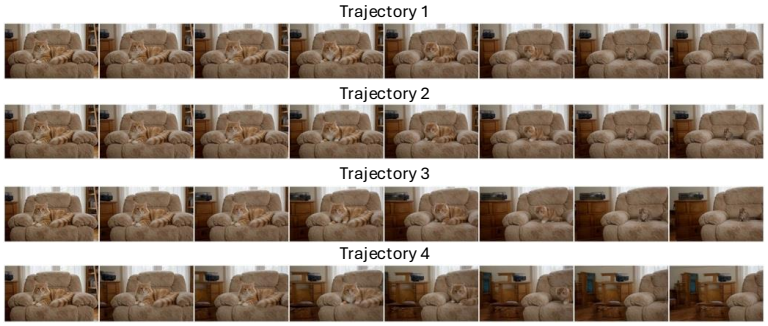


(b) Novel-view videos generated by **InspatioWorld**.

Figure 11: **Additional comparison with InSpatioWorld.** Demiurge maintains more consistent foreground details and background layout across trajectories, whereas InSpatioWorld introduces noticeable content changes.

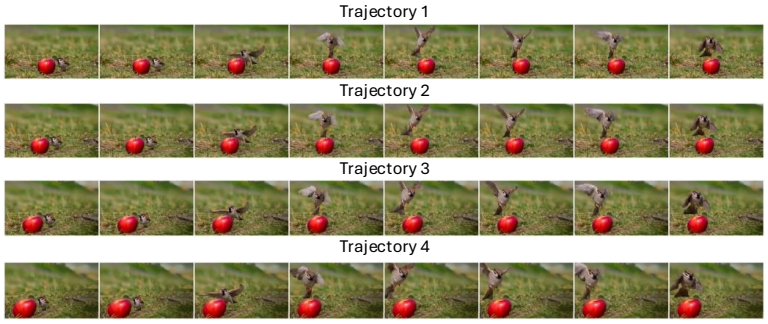


(a) Novel-view videos generated by **ours**.

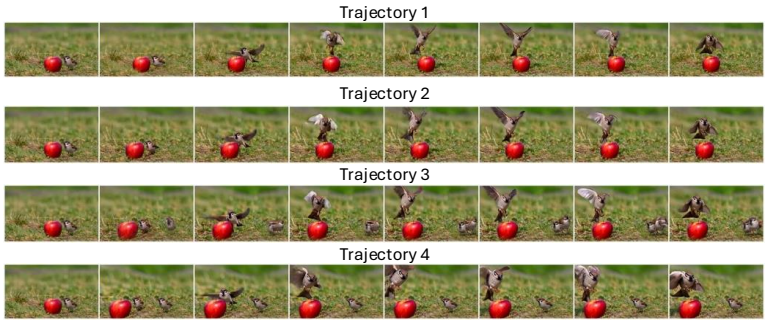


(b) Novel-view videos generated by **ReCamMaster**.

Figure 12: **Additional comparison with ReCamMaster.** Demiurge preserves object identity and spatial structure under changing viewpoints, while ReCamMaster exhibits cross-trajectory appearance and geometry inconsistencies.

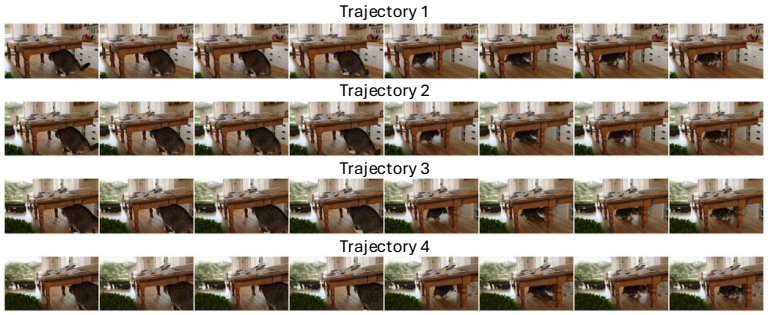


(a) Novel-view videos generated by **ours**.



(b) Novel-view videos generated by **Gen3C**.

Figure 13: **Additional comparison with GEN3C.** Demiurge produces more consistent object appearance, dynamics, and spatial relationships across multiple target trajectories.



(a) Novel-view videos generated by **ours**.



(b) Novel-view videos generated by **Spatia**.

Figure 14: **Additional comparison with Spatia.** Demiurge follows the prescribed camera trajectories while maintaining coherent scene content, whereas Spatia exhibits weaker viewpoint variation and cross-trajectory consistency.

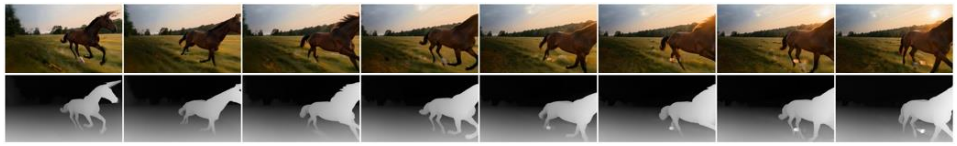


(a) Novel-view videos generated by **ours**.



(b) Novel-view videos generated by **Voyager**.

Figure 15: **Additional comparison with Voyager.** Demiurge better preserves the identity, scale, and geometry of the dynamic foreground object across substantial camera motions.



(a) Results obtained without **noise monitor.**



(b) Results obtained with **noise monitor.**



(a) Results obtained without **noise monitor.**



(b) Results obtained with **noise monitor.**

Figure 16: **Additional ablation of the noise monitor.** The noise monitor filters unreliable cache projections, reducing boundary artifacts and stabilizing the recovered geometry over time.



(a) Results obtained without **video depth completion.**



(b) Results obtained with **video depth completion.**

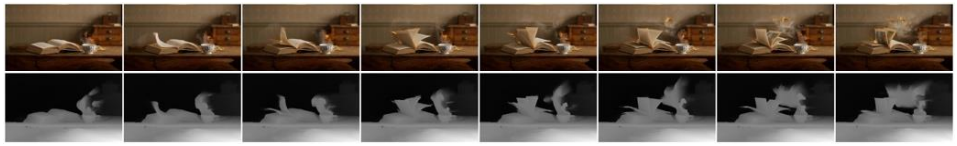


(a) Results obtained without **video depth completion.**

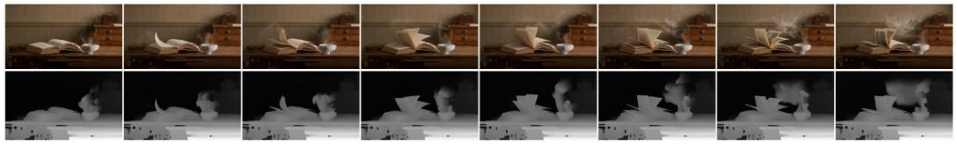


(b) Results obtained with **video depth completion.**

Figure 17: **Additional ablation of video depth completion.** Cache-consistent depth completion produces more coherent object geometry and temporal structure than direct monocular depth estimation.



(a) Results obtained without **appearance refinement.**



(b) Results obtained with **appearance refinement.**



(a) Results obtained without **appearance refinement.**



(b) Results obtained with **appearance refinement.**

Figure 18: **Additional ablation of geometry-aware refinement.** Refinement suppresses residual appearance artifacts and texture flicker while preserving the recovered geometry.



(a) Results obtained without **geometry stream warping.**



(b) Results obtained without **appearance stream warping.**



(c) Results obtained with **dual-stream stream warping.**

Figure 19: **Additional ablation of dual-stream warping.** The appearance stream preserves visual details, while the geometry stream provides reliable occlusion reasoning; combining them yields consistent appearance and geometry.



Figure 20: **Compatibility with GEN3C (Case 1, RGB).** Replacing our appearance generator with GEN3C still enables coherent RGB generation across twelve target trajectories. Frames in each row are arranged chronologically.

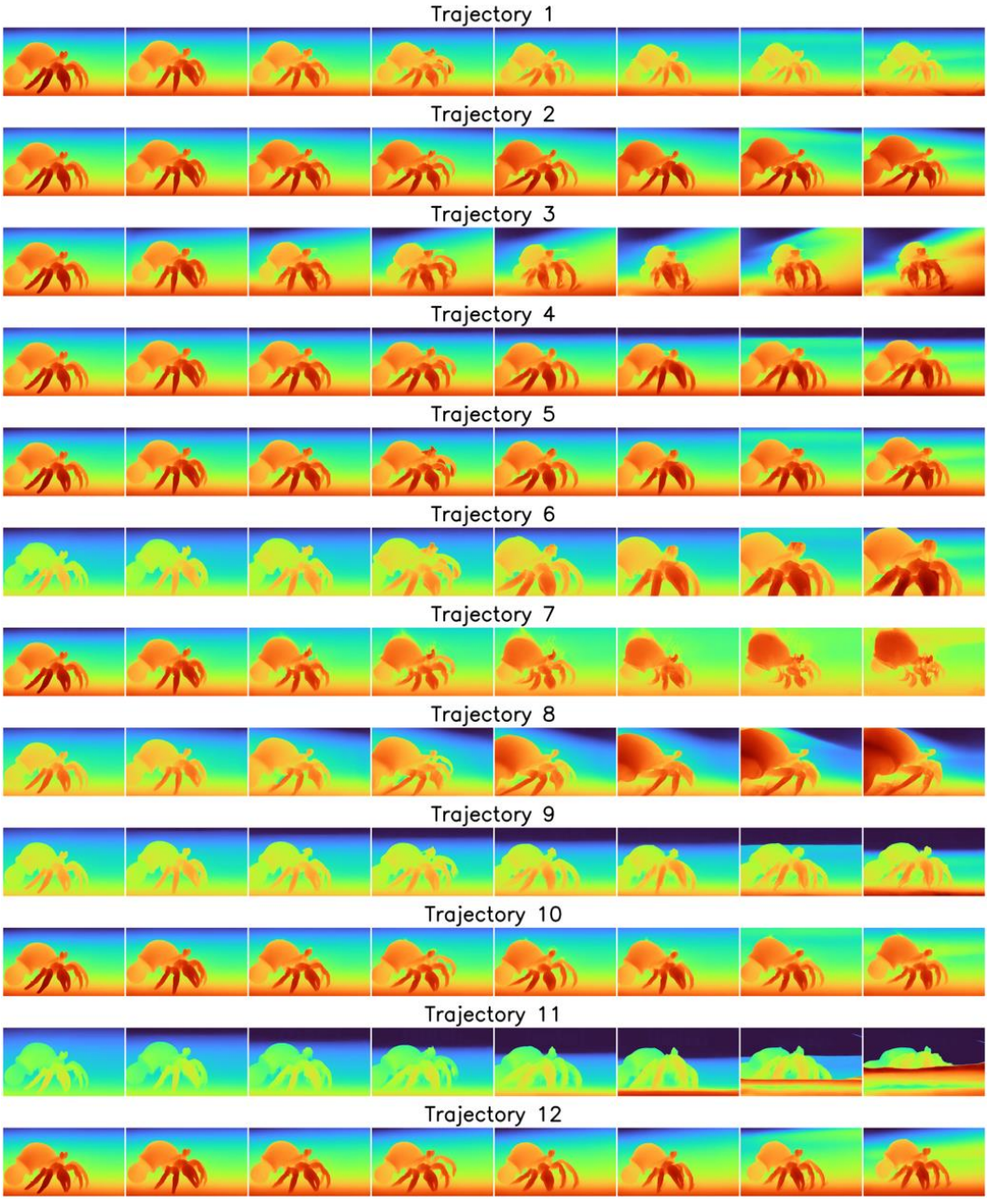


Figure 21: **Compatibility with GEN3C (Case 1, depth).** Our depth completion module recovers temporally coherent geometry from the GEN3C-generated videos for subsequent 4D memory updates.

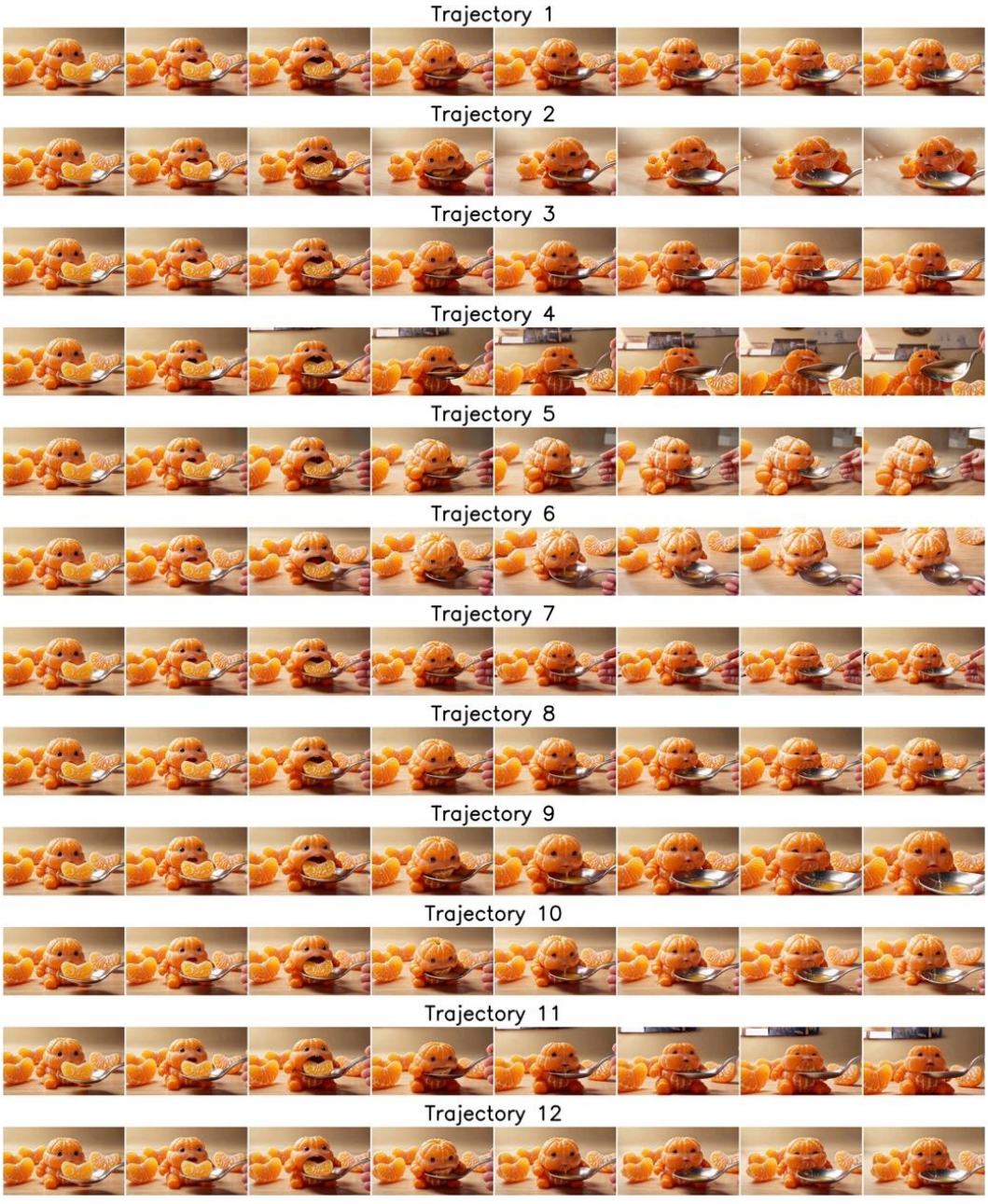


Figure 22: **Compatibility with GEN3C (Case 2, RGB).** The integrated framework preserves the dynamic subject and scene appearance across twelve target trajectories.

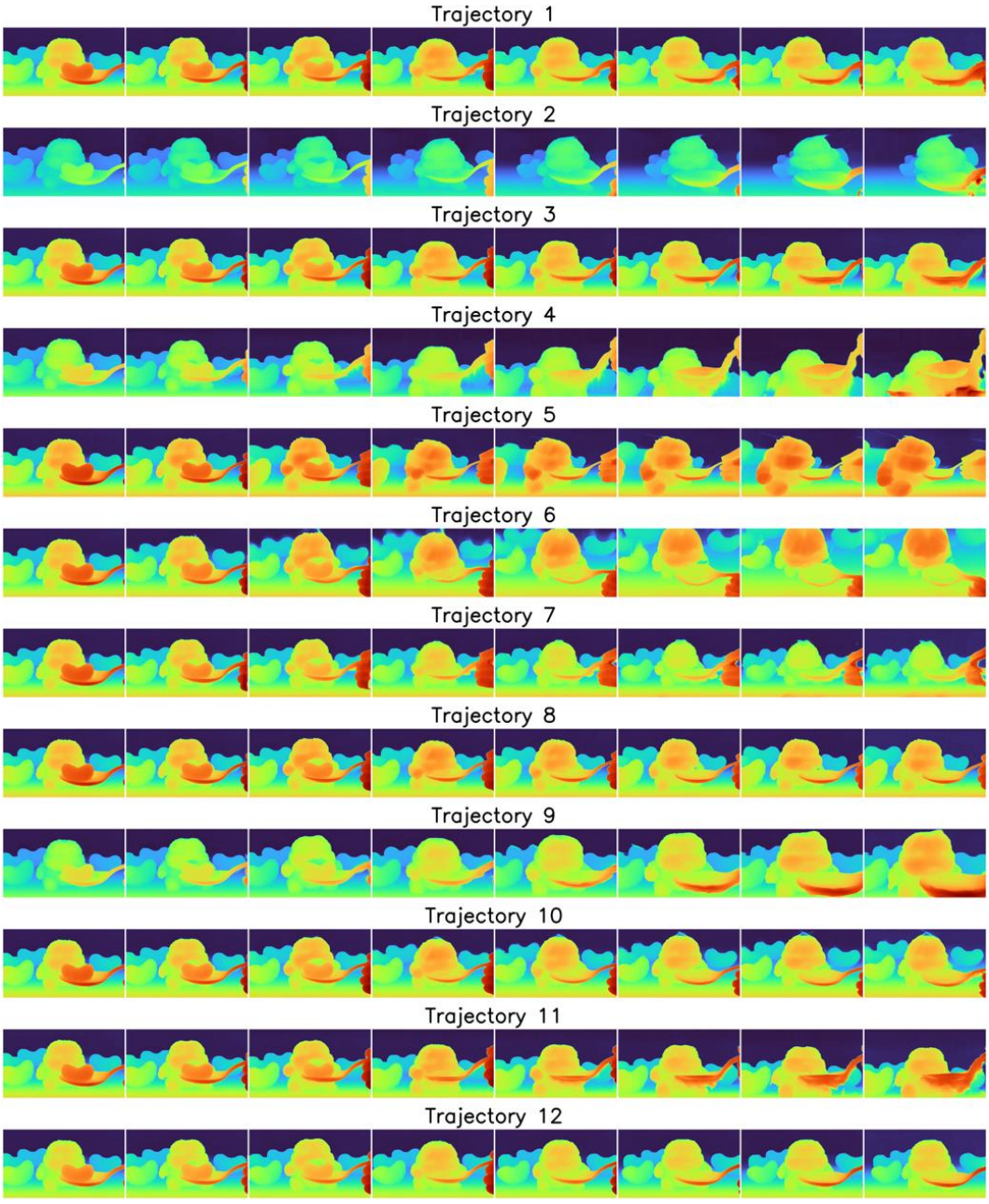


Figure 23: **Compatibility with GEN3C (Case 2, depth).** Consistent depth recovery across trajectories demonstrates that the remaining Demiurge pipeline can operate with an existing appearance generator.

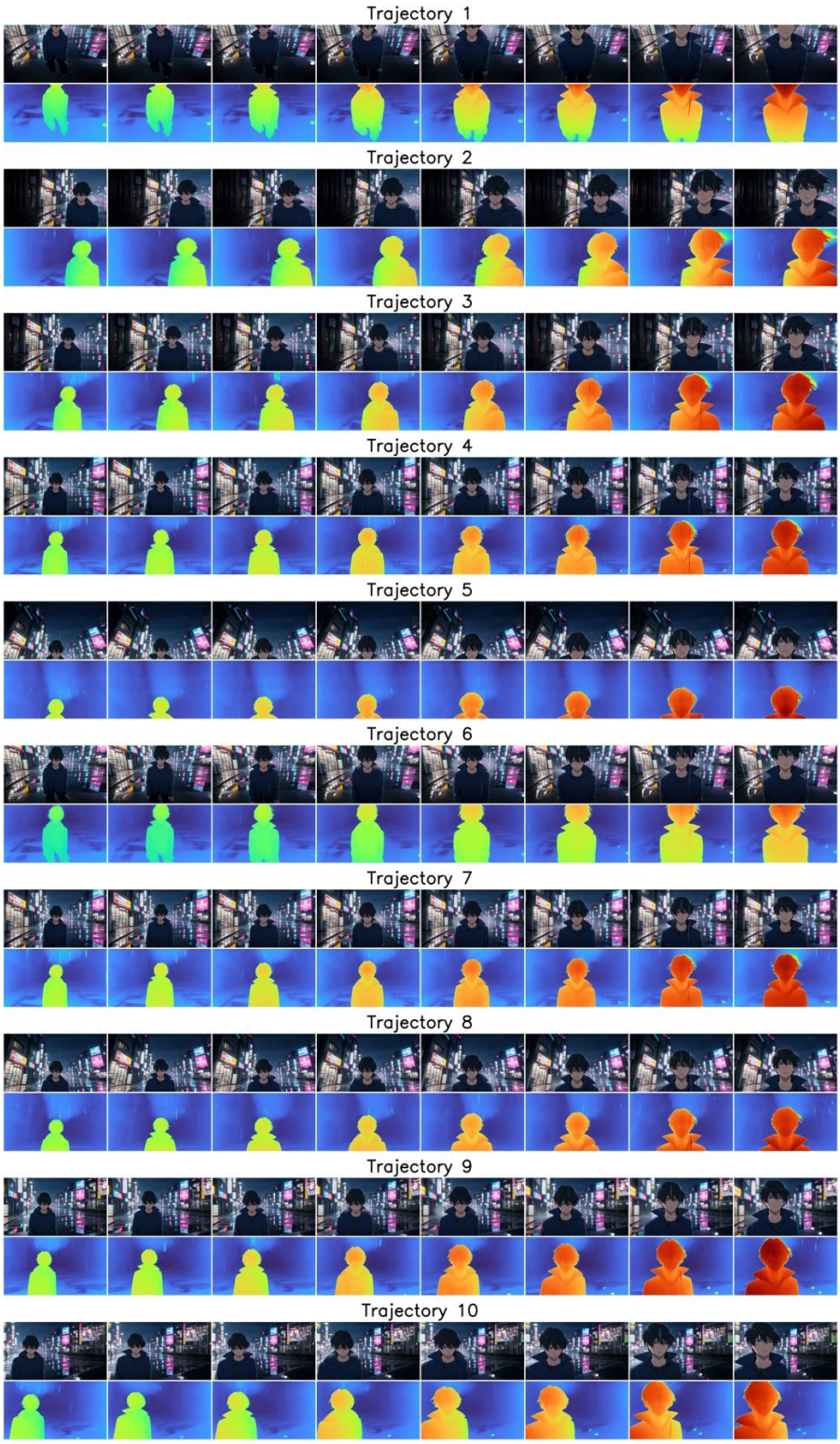


Figure 24: **Additional visualization (Case 1).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

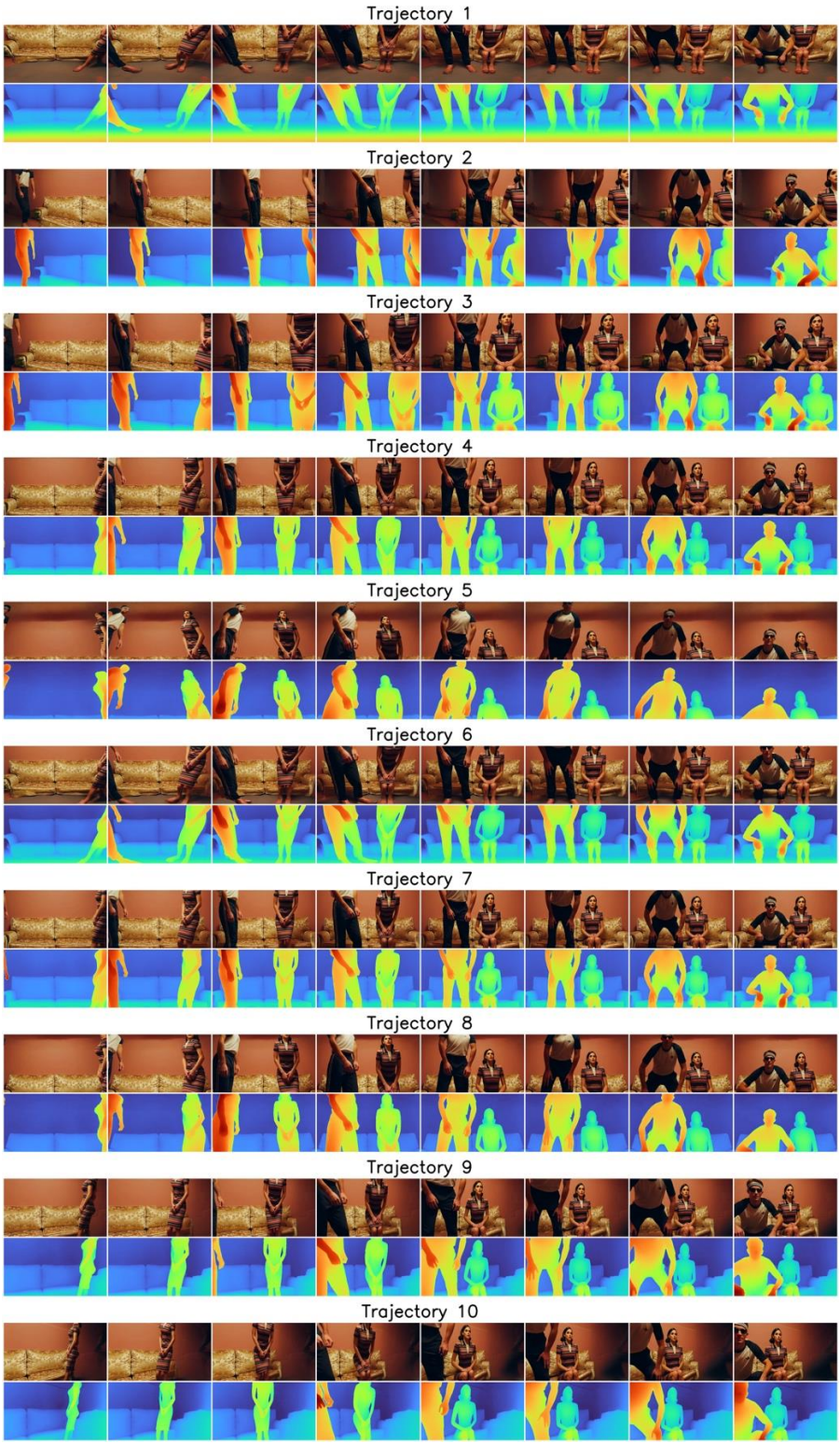


Figure 25: **Additional visualization (Case 2).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

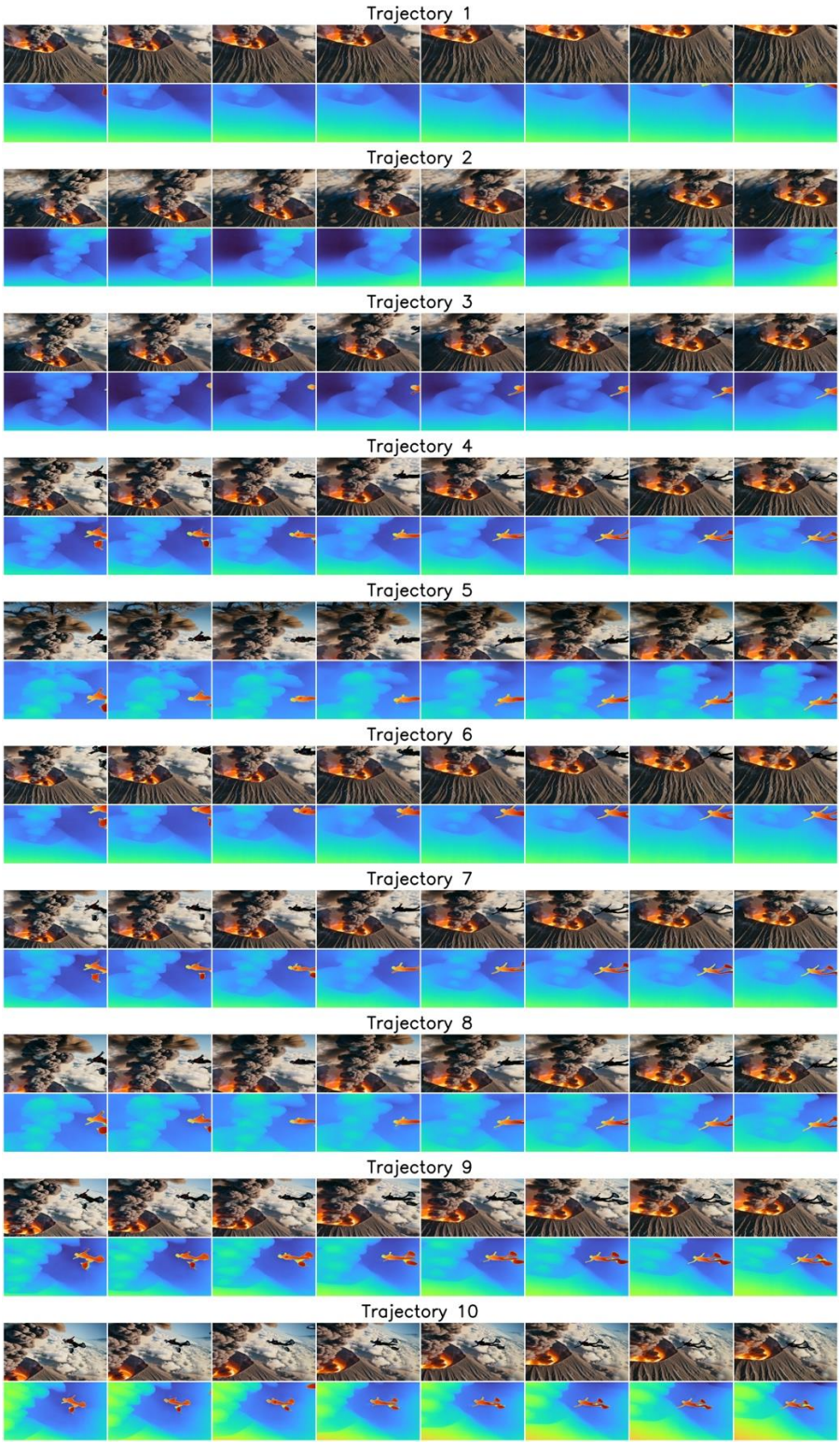


Figure 26: **Additional visualization (Case 3).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

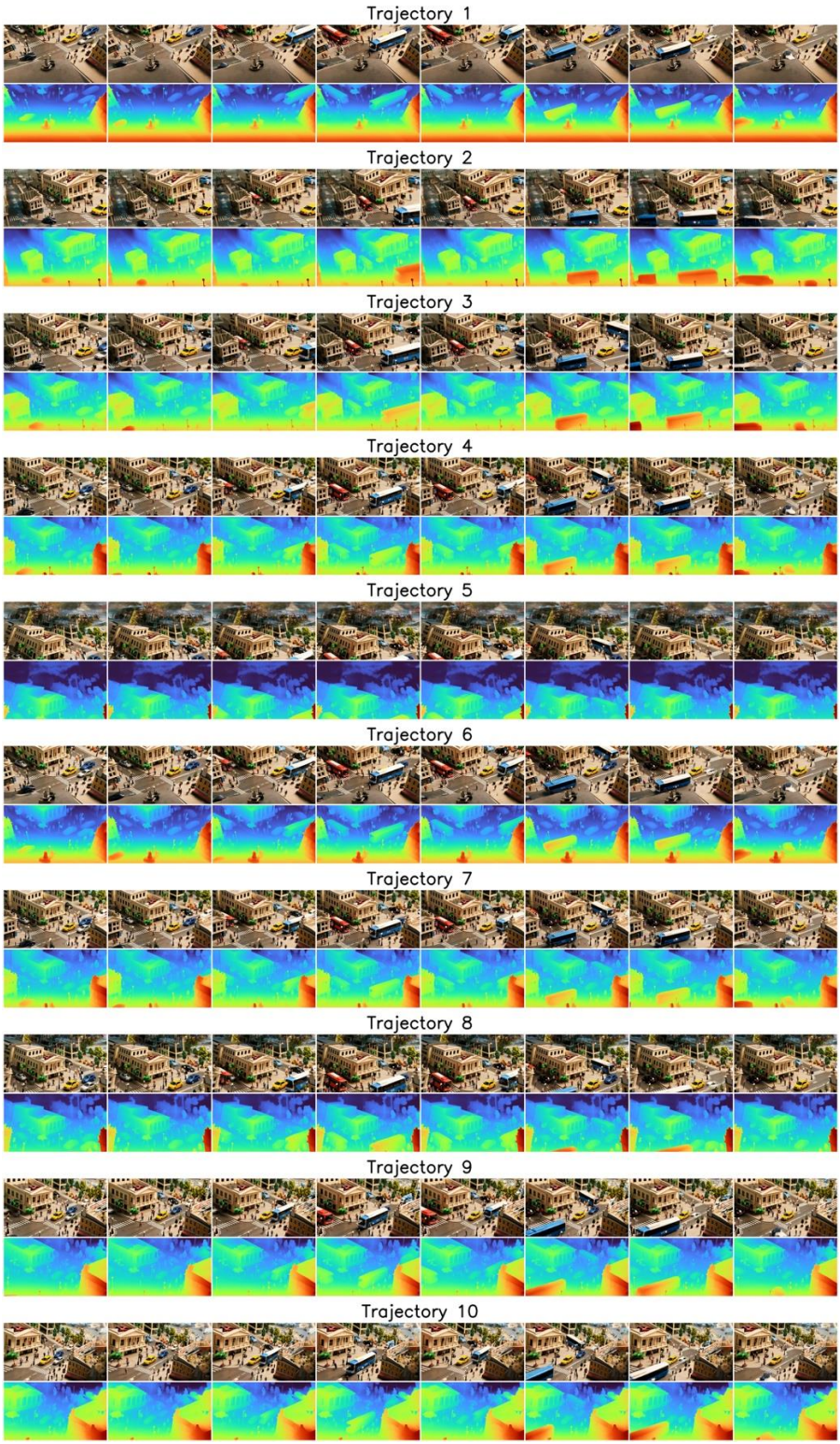


Figure 27: **Additional visualization (Case 4).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

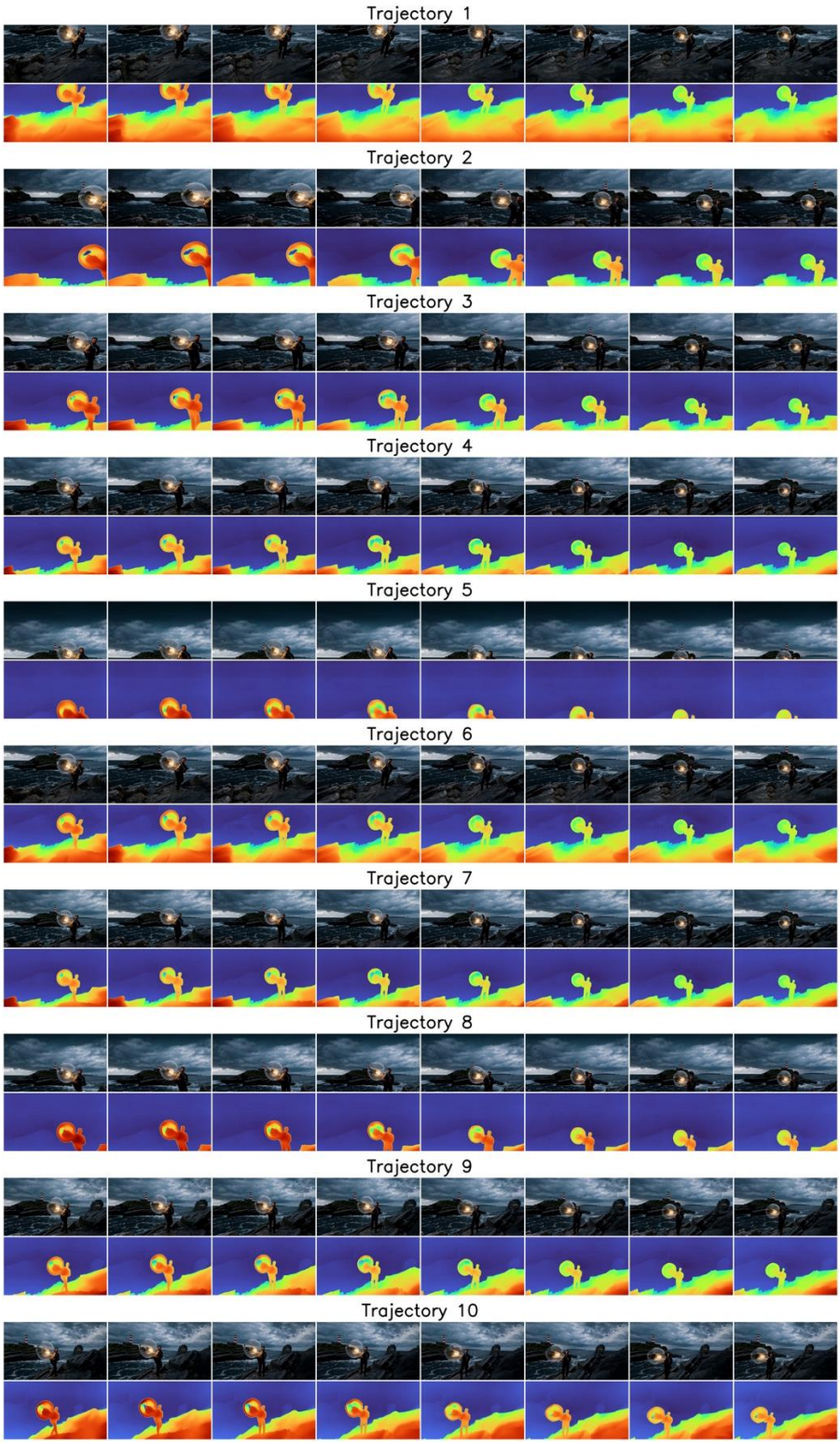


Figure 28: **Additional visualization (Case 5).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

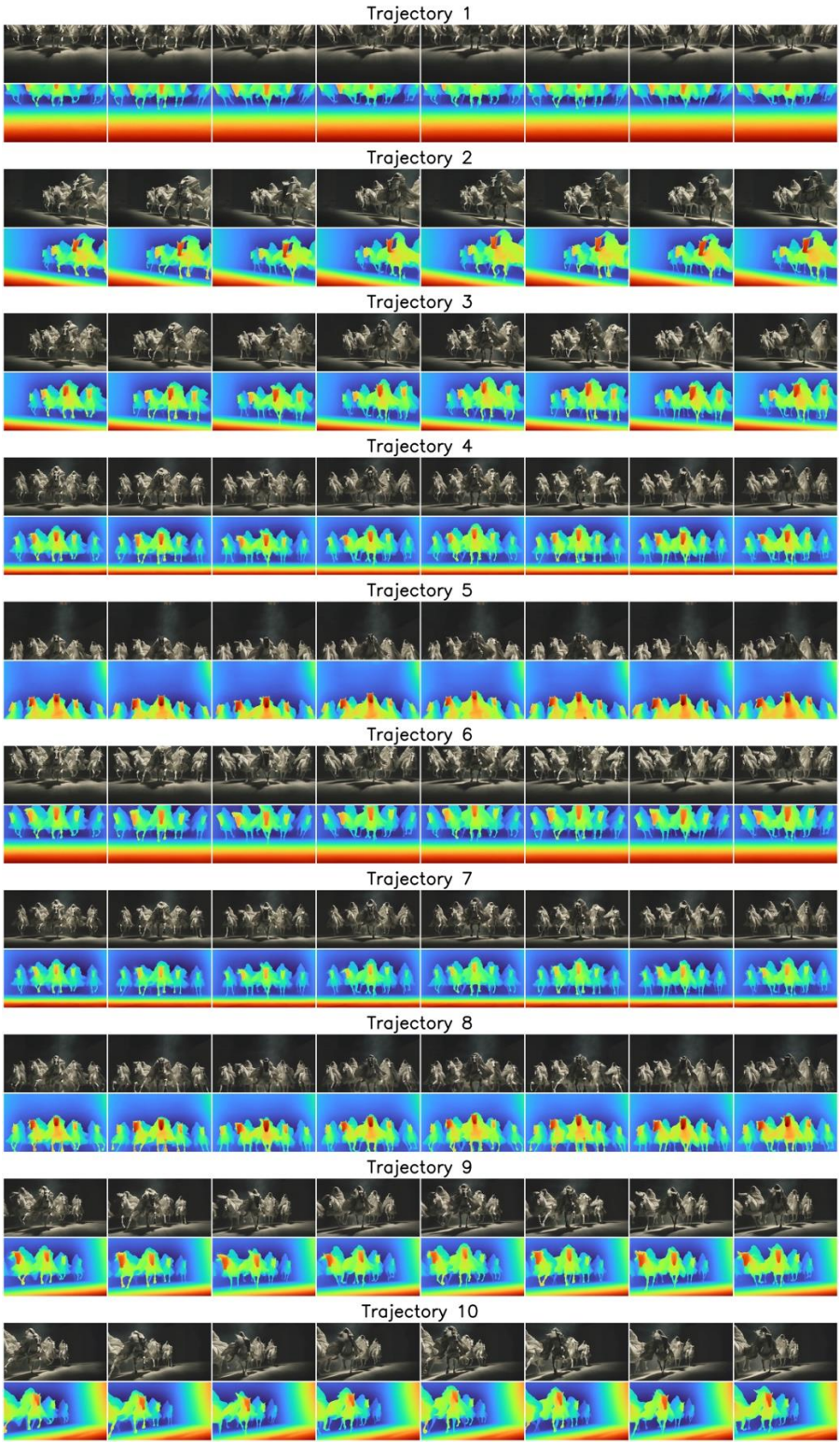


Figure 29: **Additional visualization (Case 6).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

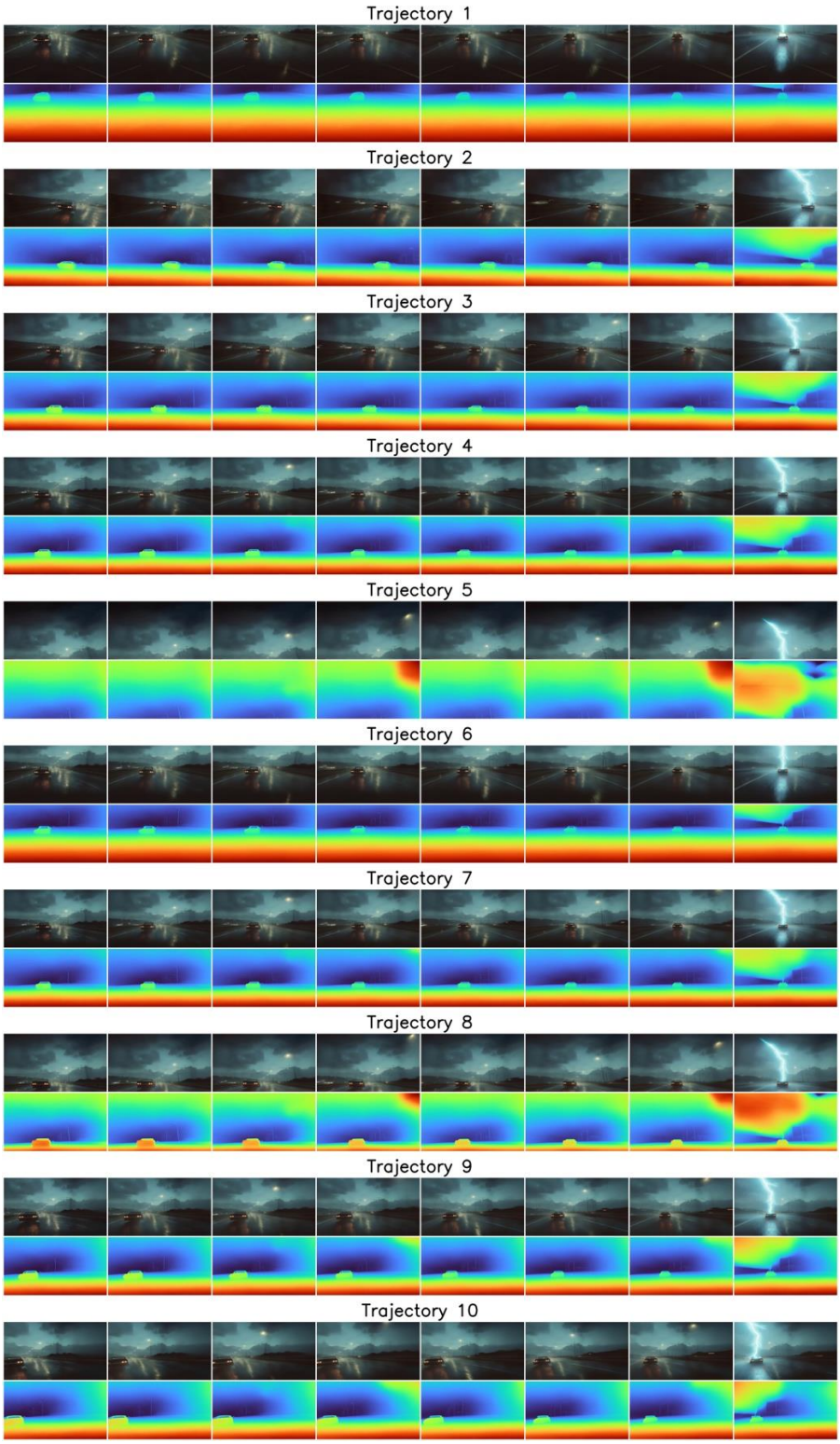


Figure 30: **Additional visualization (Case 7).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

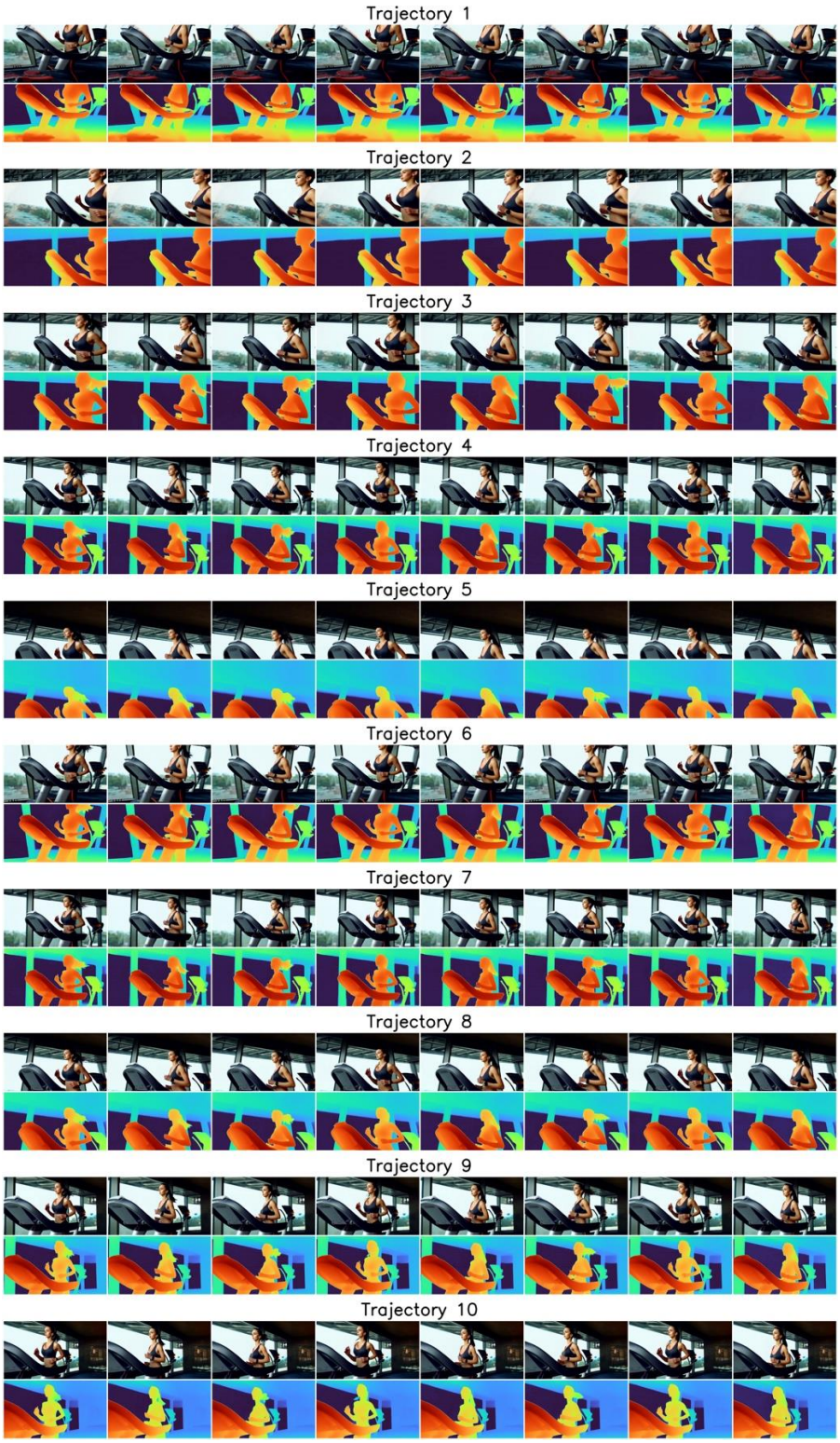


Figure 31: **Additional visualization (Case 8).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.

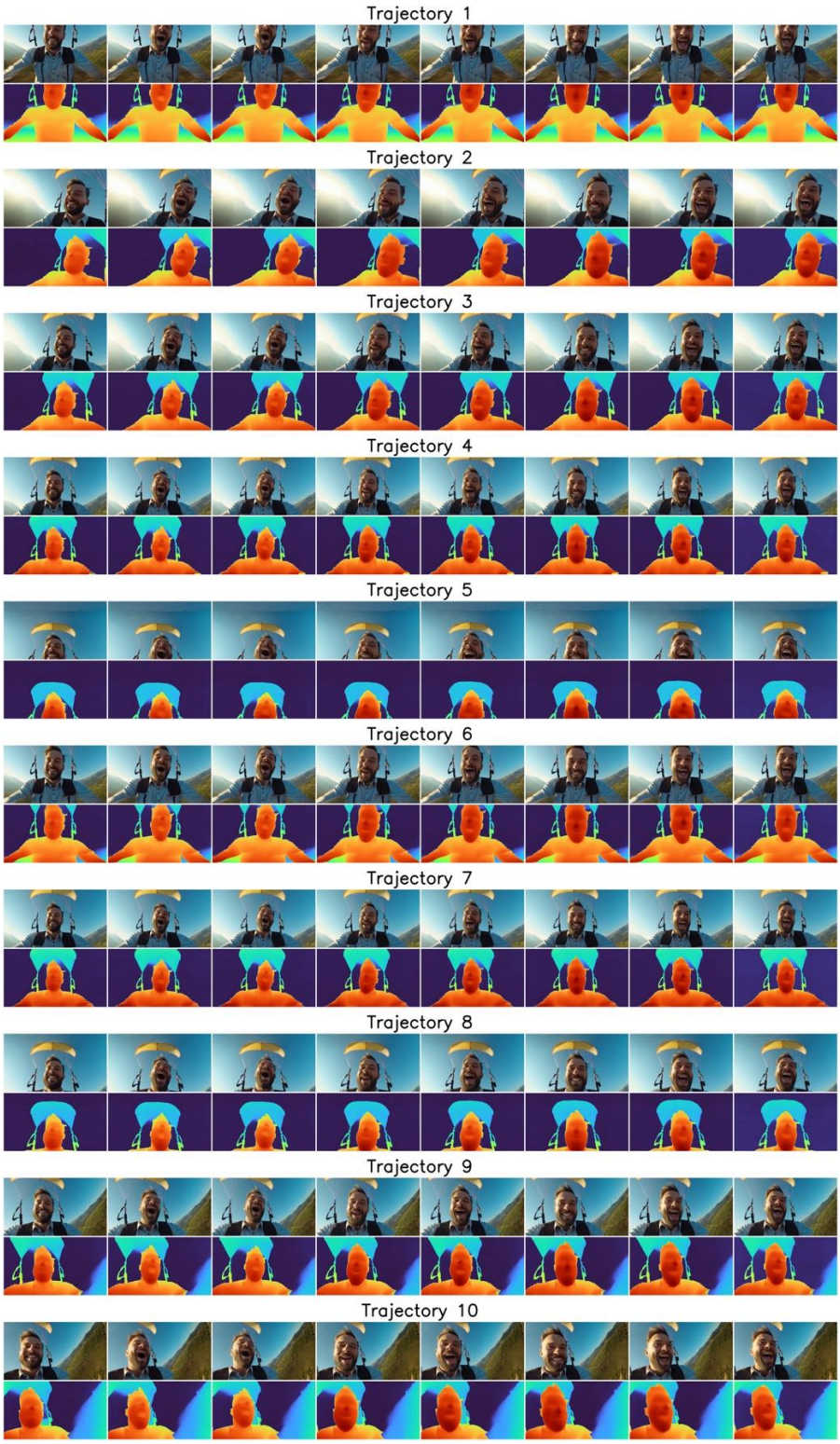


Figure 32: **Additional visualization (Case 9).** Generated RGB frames and recovered depths across ten target camera trajectories. Each row corresponds to one trajectory, with sampled frames arranged chronologically from left to right.