Persistent Planar Memory for Video World Models

Yuze He[1,2] **Bin Tan**[2] **Zelin Gao**[3,2] **Yujun Shen**[2]
Yong-Jin Liu[1,*] **Nan Xue**[2,*] **Yinghao Xu**[3,2]
[1]Tsinghua University [2]Ant Group [3]HKUST

Abstract

Video world models that generate and explore environments at scale need a persistent 3D memory of what they have produced. Plane primitives offer a natural candidate: a compact set of oriented planes captures the dominant structural elements (*e.g.,* walls, floors, facades) of indoor and outdoor scenes, while requiring only 7 parameters per primitive. We present **PPMem**, a video world model that adopts plane primitives as its persistent 3D memory. PPMem augments a video diffusion transformer with a planar prediction head and a Depth Token Fusion (DTF) module: each autoregressive chunk jointly produces RGB video and planar geometry, which merges into a persistent plane memory that conditions subsequent chunks via rendered depth. The resulting memory is three orders of magnitude more compact than per-pixel point clouds ($\sim$100K primitives, $\sim$5 MB for a full scene), geometrically explicit (metric depth, normals, exportable surfaces), cross-view consistent (jointly predicted by the shared backbone), and adaptive (refined by the merging procedure as generation proceeds). On DL3DV and RealEstate10K, PPMem outperforms prior 3D-grounded world models in video quality while jointly producing metric-scale depth, surface normals, and exportable planar geometry, sustaining coherence across hundreds of frames under large viewpoint changes. *Project page:* `https://hyzcluster.github.io/ppmem`.

1 Introduction

Humans build the mental representation of their world through structural abstraction, decomposing scenes into points, lines, and planes. Among the three, planes carry disproportionate weight in human-made environments: walls, floors, ceilings, tabletops, building facades, and ground surfaces [2, 3] can all be approximated by only a small set of planes. A model that generates entire worlds, not just short clips, needs an analogous compact, structured memory of what it has produced. Plane primitives are a natural candidate for this role. Each primitive is parameterized by just 7 numbers (a center depth, two rectangular extents, and a rotation quaternion), which explicitly represents 3D geometric structure, and is orders of magnitude more compact than per-pixel representations. This raises a natural question: *can plane primitives serve as a video world model's representation, acting as a compact yet expressive persistent memory?*

Video world modeling has emerged as a central frontier of AI, with systems that generate, simulate, and explore environments at scale. Existing approaches roughly fall into two camps. **Pure-video world models**, such as Genie 3 [6] and LingBot-world [5], train end-to-end on massive video corpora and rely on implicit attention and scale to maintain coherence. They achieve remarkable visual fidelity but expose no explicit 3D state for downstream conditioning, editing, or export. **3D-grounded video world models**, the direct comparison group for PPMem, augment diffusion with explicit 3D geometric memory: a 2-frame point-cloud buffer (Gen3C [24]), an oriented-surfel store (VMem [13]), a TSDF volume paired with a keyframe pool (SPMem [37]), or a raw-frame retrieval bank (Context

*Corresponding authors.

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



Figure 1: **Given a single input image and a camera trajectory, PPMem autoregressively generates video while accumulating a persistent plane-based 3D memory.** We show two DL3DV scenes (indoor, outdoor) with generated frames at selected indices. Insets visualize the plane pool at each frame, colored by the current view frustum. The plane geometry grows to cover the full explored scene, serving as persistent spatial memory for subsequent generation.

as Memory [41], WorldMem [38]). A related line, Voyager [11], jointly generates RGB and depth video and re-projects the generated depth into a point-cloud cache for conditioning subsequent chunks. Yet no existing 3D-grounded representation simultaneously exhibits the four properties that an explicit, persistent 3D state demands: **compactness** (orders of magnitude fewer primitives than dense representations), **geometric explicitness** (metric, rasterizable, exportable), **cross-view consistency** (jointly predicted by the generation backbone rather than assembled per frame), and **adaptivity** (refining the representation as generation proceeds).

We present **PPMem**, a video world model that uses plane primitives as its underlying 3D structure, accumulating them in a **persistent plane memory** for long-context video generation. Concretely, PPMem augments a video diffusion transformer (DiT) with two additions: a planar prediction head adapted from Plana3R [18] that regresses plane parameters per chunk, and a **Depth Token Fusion** (DTF) module that re-injects depth rendered from the memory into the DiT features. The memory plays a dual role: as the model's structured 3D output (metric-scale depth, surface normals, and exportable planar geometry), and as persistent spatial conditioning for subsequent generation. Plane primitives are particularly well-suited for this role: after merging, $\sim$100K planes cover a full scene three orders of magnitude more compactly than per-pixel point clouds (Table 1), and are fully differentiable through rectangular Gaussian rasterization [18]. Under large viewpoint changes, PPMem sustains coherent, metric-grounded video generation across hundreds of frames (Figure 1).

Compared with Gen3C, Voyager, VMem, and SPMem, PPMem achieves state-of-the-art video quality on DL3DV and RealEstate10K while accumulating under 10 MB of 3D memory.

Our contributions are:

- We propose plane primitives as a 3D representation for video world models, accumulated in a persistent plane memory via cross-chunk geometric merging — to our knowledge, the first such memory that is simultaneously compact, geometrically explicit, cross-view consistent, and adaptive.
- We instantiate PPMem on a 5B Wan-2.2 DiT with a Depth Token Fusion (DTF) module that re-injects rendered plane-pool depth into the generation process, enabling joint generation of RGB, metric depth, surface normals, and planar geometry in a single framework.

- We benchmark PPMem against Gen3C, Voyager, VMem, and SPMem on short-context and long-context evaluations across various datasets, demonstrating sustained visual and geometric coherence across hundreds of frames under large viewpoint changes.

2 Related Work

Camera-controlled video generation. Recent video diffusion models with camera control [24, 1, 42, 7, 8, 35, 11] generate impressive short clips of 14–121 frames. However, extending generation to longer sequences reveals a *representation gap*: most methods either lack a persistent 3D state entirely (relying on camera embeddings or frame-level conditioning) or maintain only minimal geometric state. We organize prior work below by the *level of 3D representation* they maintain, highlighting a progressive trend toward more structured representations.

From depth maps to point clouds. Depth estimation provides geometric cues for video world models [40]. At the simplest level, Gen3C [24] maintains a 2-frame point buffer (the seed frame and the most recent chunk-boundary frame, whose monocular depth is re-estimated and aligned to the cache at each autoregressive step); these 3D points are forward-warped as conditioning for the next generation chunk. This provides per-chunk geometric grounding but carries no persistent structure beyond two frames: all intermediate geometry is overwritten. Sora3R [22] predicts dense 4D pointmaps but is restricted to short sequences. At this level, depth and point representations are per-pixel and unstructured, offering no abstraction beyond raw geometry.

Surfels, TSDF, and frame-level memory. A step up in structure, VMem [13] reconstructs oriented surfels via CUT3R [31] during autoregressive generation and uses them to index relevant past views; however, the surfel count grows with each generated frame. SPMem [37] builds a TSDF volume as spatial memory alongside an episodic keyframe pool; extending to long sequences requires re-estimating depth and re-integrating all accumulated frames into the volume at each step, with cost scaling linearly in sequence length. Context as Memory [41] retrieves raw past frames by FOV overlap, and WorldMem [38] uses cross-attention over a frame bank with relative Plücker embeddings; both operate at the frame level without maintaining explicit 3D geometry.

Joint depth and 3D generation. A complementary line of work produces geometry alongside or from video. Voyager [11] jointly generates video and depth via shared diffusion and proposes re-projecting generated depth into a point-cloud cache for extended generation; its conditioning depth is normalized per-frame to relative disparity. IDCNet [21] outputs metric depth with camera control. GeometryCrafter [39] is a post-hoc tool that estimates temporally coherent point maps from *existing* video without generating new content. ViewCrafter [43] renders an externally estimated point cloud (DUSt3R [34]) at target viewpoints and uses diffusion to inpaint holes, consuming 3D rather than predicting it. FlashWorld [14] extends Wan 2.2 [29] with a 3D Gaussian decoder for joint multi-view generation, but repurposes the temporal dimension as a multi-view axis to produce *static* 3D scenes rather than temporal video sequences. None of these methods accumulate a compact, structured 3D representation that serves as persistent memory for long-context video generation.

Plane-based 3D reconstruction and our position. Learning-based plane detection progresses from single-view segmentation (PlaneNet [19], PlaneRCNN [20], PlaneTR [26]) to differentiable multi-view reconstruction: AlphaTablets [9] optimizes alpha-composited planar primitives from monocular video, and PlanarSplatting [27] extends 3D Gaussian splatting [36] with planar constraints. Plana3R [18] performs *feed-forward* planar splatting via differentiable rectangular Gaussian rasterization, predicting metric-scale plane primitives from unposed image pairs in a single pass without per-scene optimization. All of these methods are *reconstruction-only*: they extract geometry from already-observed images. PPMem repurposes planar prediction as *persistent spatial memory within a generative model*: planes are jointly predicted alongside each generated video chunk by a shared diffusion backbone, and the accumulated plane pool feeds back as geometric conditioning for subsequent chunks. This bidirectional loop (generation $\rightarrow$ planes $\rightarrow$ conditioning $\rightarrow$ generation) has no counterpart in prior plane reconstruction work. We compare against alternative 3D representations in §3.1.

Table 1: **Properties of 3D representations for cross-chunk video memory.** We compare along four aspects: whether geometry is predicted *internally* by the generation model or externally, whether predictions are cross-view consistent, the number of persistent 3D primitives, and whether the representation captures structured surfaces. Primitive counts are estimates at $\sim$400 frames, 576×1024.

	Dense PC [24]	Surfels [13]	TSDF→PC [37]	**Planes (PPMem)**
3D source	External (MoGe [32])	External (CUT3R [31])	External (DA3 [15])	**Internal (joint head)**
Cross-view consistent	✗	✗	partial	✓
Primitives (accumulated)	$\sim$210M†	$\sim$250K	$\sim$1–2M	$\sim$**100K**
Structured surfaces	✗	✗	✗	✓

†Theoretical total if all per-pixel points were accumulated.

3 PPMem: Persistent Planar Memory

PPMem uses plane primitives as a compact, persistent 3D memory for autoregressive video generation. Unlike point clouds, surfels, or TSDF volumes used by prior 3D-grounded world models, plane primitives are predicted *internally* by the generation backbone (requiring no external depth estimator), yield cross-view consistent geometry by construction, and compress an entire scene into $\sim$100K primitives — three orders of magnitude fewer than dense point clouds (Table 1). Each generated chunk jointly predicts video and a set of oriented rectangles in 3D, which are accumulated into a global pool via geometric merging; the pool is then rendered and fed back as depth conditioning for subsequent chunks. We describe the representation and memory mechanism (§3.1), the model architecture (§3.2), and training (§3.3) below.

3.1 Plane Primitives and Persistent Memory

A plane primitive in our framework is a parametric oriented rectangle in 3D space, defined by 7 parameters: a *center depth* z along the camera ray (from which the 3D center $\mathbf{x}$ is derived via the camera intrinsics), two *rectangular extents* (s_x, s_y), and a *rotation quaternion* $q \in \mathbb{R}^4$ whose third rotation-matrix column gives the surface normal $\mathbf{n}$. Each primitive is rendered via differentiable rectangular Gaussian rasterization [18], producing depth and normal maps. At inference, predicted primitives are transformed to world coordinates and accumulated into the persistent plane pool, which after merging typically contains $\sim$100K primitives ($\sim$5 MB) representing the full explored scene. Beyond compactness and internal predictability (Table 1), plane parameters can be merged across autoregressive steps via simple geometric criteria (occlusion voting, coplanarity, distance), making them well-suited as a cross-chunk memory.

Persistent plane pool. The persistent plane pool $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots\}$ is a global, accumulating set of plane primitives across autoregressive chunks. After each chunk produces predicted planes $\mathcal{P}_{\text{new}}$, we merge them into $\mathcal{P}$ via the procedure below. Before the next chunk, we render $\mathcal{P}$ at the target camera poses and feed the rendered depth as conditioning to the generation model (detailed in §3.2). The full inference loop is summarized in Algorithm 1 (Appendix).

Plane pool maintenance. After each chunk, new planes are merged into $\mathcal{P}$ with two goals: remove erroneous pool planes exposed by newer observations, and add new planes only where they contribute genuine new geometry. Each plane carries an **observation quality score** $s = r/|\cos\theta|$, where r is the Euclidean distance to the creating camera and θ is the angle between the viewing ray and the surface normal (lower is better: the camera is closer and views the surface more perpendicularly). Three mechanisms maintain pool quality:

1. **Occlusion deletion** (*correcting the past*): pool planes that float in front of newly predicted surfaces are flagged via multi-camera voting and deleted.
2. **Bidirectional quality dedup** (*improving the present*): among nearby planes (within distance ϵ), only the one with the better quality score survives — both within a chunk (*intra-step*, typically reducing count by $\sim$80%) and across chunks (*cross-step*).
3. **Forward-looking prune** (*anticipating the future*): pool planes that will be viewed from much closer in the next chunk ($r_{\text{next}}/r_{\text{created}} < \tau$) are removed proactively, as the upcoming

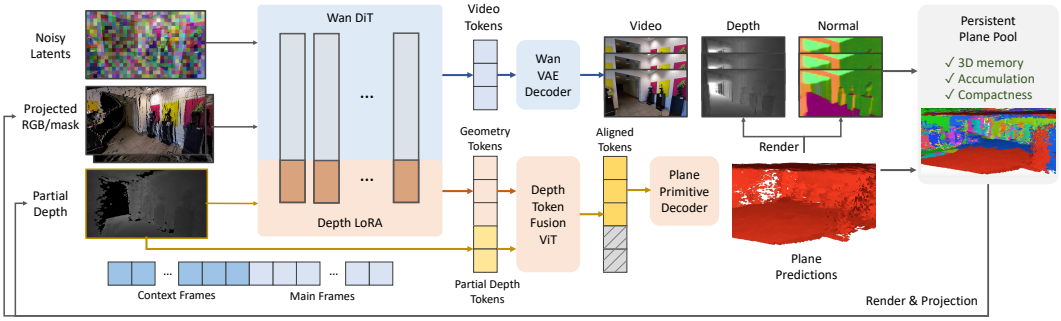


Figure 2: **PPMem architecture overview.** The Wan DiT receives noisy video latents channel-concatenated with projected source-view RGB/masks and partial depth rendered from the plane pool; context frames are prepended temporally. The DiT produces two token streams: *video tokens* decoded by the Wan VAE into RGB outputs, and *geometry tokens* adapted by geometry LoRA and fused with partial depth tokens via a ViT-based Depth Token Fusion module. The fused tokens are decoded into per-frame plane primitives, which merge into the persistent plane pool. The pool is rendered at the next chunk's target cameras and fed back as conditioning.

chunk will produce higher-quality replacements. This uses only the next chunk's camera poses, which are supplied as camera-control inputs rather than predicted.

Additional filtering stages are detailed in Algorithm 2 (Appendix). Surviving new planes are appended with metadata. Merging is local and does not require gradient-based optimization.

Pool-to-chunk conditioning. Before each new chunk, we render $\mathcal{P}$ at the target camera poses via differentiable rasterization, producing partial depth maps that condition the next chunk through two complementary paths: VAE-encoded as a latent channel-concatenated with the noisy video input, and at pixel resolution through token-level fusion (§3.2). Unlike the fixed-size buffers of prior work, $\mathcal{P}$ accumulates geometry across *all* past chunks; its growth is controlled by merging, typically reaching $\sim$100K primitives ($\sim$5 MB) after $\sim$400 frames.

3.2 Model Architecture

PPMem builds on the Wan 2.2 video diffusion transformer [29] with two architectural additions (Figure 2): a **plane prediction head** that regresses plane primitives from DiT features, and a **Depth Token Fusion (DTF) module** that injects rendered plane-pool depth as token-level conditioning. At inference, PPMem generates video in $\sim$25-frame chunks: target camera poses are used to project source views and render the plane pool into geometric conditioning (projected RGB, visibility masks, and depth maps), which are channel-concatenated with the noisy latent and fed to the DiT. The camera is encoded entirely through projected views rather than explicit parameters, decoupling the model from specific intrinsics or trajectory scales.

Plane prediction head. The head, adapted from Plana3R [18], regresses the 7 plane parameters (§3.1) from DiT features. Since the DiT operates on a coarsely-patchified latent grid ($H/16 \times W/16$), we first apply a **feature decoder** that upsamples by $2\times$ in the spatial dimension. The decoder is initialized from the Wan 2.2 VAE decoder's upsampling block, leveraging learned spatial priors [22]. Three small MLPs then regress center depth (in log-space over $[0.2, 277]$ m), rectangular extents (in metric units), and rotation quaternion at each feature location. The predicted primitives are rendered via differentiable rectangular Gaussian rasterization [18] and supervised against reconstructed metric depth (§3.3).

Depth token fusion (DTF). To inject the plane pool's geometric context into plane prediction, we use a DTF module (inspired by MDM [28]) in the geometry pass (§3.3). Partial depth maps rendered from $\mathcal{P}$ are tokenized via a 16×16 patch convolution and processed by a small ViT-Base encoder ($\sim$87 M parameters, $\sim$1.7% of the 5B DiT), initialized from DINOv2-Base [23]; the encoded depth tokens are then concatenated with DiT feature tokens for fusion. Since rendered depth maps

contain holes (unseen regions) and partially covered patches, we adopt a **three-tier masking** strategy before fusion: (i) *fully invalid* patches (zero coverage) are always dropped; (ii) *mixed* patches (partial coverage) are dropped with probability $\rho_{\text{mix}}=0.75$; (iii) *fully valid* patches are randomly dropped to reach a target overall mask ratio in $[0.6, 0.9]$. This forces the DiT to remain robust to depth-conditioning sparsity while still benefiting from reliable conditioning signal.

3.3 Training

We train PPMem in two stages. **Stage 1** fine-tunes the Wan 2.2 backbone for camera-controlled video generation, conditioning on projected source views and context frames with the standard flow-matching diffusion loss [17]. **Stage 2** freezes the video backbone and attaches the plane prediction head, DTF module, and feature decoder; training proceeds exclusively with a geometry loss:

$$\mathcal{L} = \underbrace{\lambda_d \mathcal{L}_{\text{depth}} + \lambda_n \mathcal{L}_{\text{normal}}}_{\text{per-patch}} + \underbrace{\lambda_d^r \mathcal{L}_{\text{depth}}^{\text{rend}} + \lambda_n^r \mathcal{L}_{\text{normal}}^{\text{rend}}}_{\text{rasterization}} + \lambda_s \mathcal{L}_{\text{smooth}}, \tag{1}$$

where the per-patch terms supervise depth (L_1, log-space) and normals (cosine + L_1) per spatial token, the rasterization terms compare rendered planes against ground truth (enabled after 1000 warmup iterations), and $\mathcal{L}_{\text{smooth}}$ is a total-variation regularizer on rendered depth. Default weights: λ_d=0.2, λ_n=0.05, λ_d^r=0.02, λ_n^r=0.01, λ_s=0.02 (full configuration in Appendix A.2). We inject LoRA adapters [10] on self-attention in all 30 DiT blocks, providing geometry-specific attention patterns atop the frozen video-generation weights. At inference, video and geometry are fully decoupled: all denoising steps run with the frozen Stage-1 backbone (LoRA disabled), producing video at the original model's quality. A single additional forward pass with LoRA enabled extracts plane geometry from the final-step features. Stage 1 runs for $\sim$7 000 iterations and Stage 2 for $\sim$5 000.

4 Experiments

4.1 Setup

Training data. We train on a mixture of DL3DV [16] and RealEstate10K [44] (80K+ clips) at 576×1024 resolution. For ablation experiments (Tables 3–5), we train on a DL3DV indoor subset ($\sim$5K clips) at 288×512. Both settings use metric depth from MoGe-2 [33], scale-aligned to VGGT [30] reconstructions (Appendix A.1).

Test sets. We evaluate on 100 DL3DV test scenes and 100 RealEstate10K (RE10K) clips (both disjoint from training); ablations use the DL3DV indoor subset (56 scenes). For long-context evaluation, we use the full DL3DV test sequences (300–400 frames each), which exhibit substantially larger camera motion than RE10K and stress the persistent memory mechanism.

Baselines. We compare against Gen3C [24], Voyager [11], VMem [13], and SPMem [37], all run with their public release code and default hyperparameters. All methods receive ground-truth camera trajectories and first-frame depth (or Plücker rays for VMem).

Protocol. Each method generates at its own native resolution and aspect ratio; input images and ground truth are center-cropped and resized to match, so that metrics reflect each method's best operating point. We report short-range results at N=49 and N=121 frames (Table 2).

Metrics. For short-range evaluation (Table 2), we report PSNR, LPIPS, CLIP similarity, and DINOv2 similarity. For long-context evaluation (Figure 3), we report CLIP and DINOv2 similarity as a function of frame index, which capture semantic and structural fidelity across large viewpoint changes. Per-frame depth accuracy against baselines is shown in Figure 5(a); depth metrics — absolute relative error (AbsRel = $|d - d^*|/d^*$), threshold accuracy (δ<1.25), and mean normal angle error (Normal°) — are additionally reported in the ablation tables.

Implementation. PPMem is built on Wan 2.2 TI2V 5B [29]. We use a per-frame VAE encoder (no temporal compression, 1:1 latent-to-frame mapping) to preserve per-frame geometric correspondence for plane prediction. Depth conditioning is log-normalized over $[0.2, 99]$ m, while the plane head predicts log-depth over $[0.2, 277]$ m. We train with learning rate 1×10^{-5}, classifier-free guidance dropout 0.1, and context-block dropout 0.1 (all context frames dropped together). Full hyperparameters are in Appendix Table S1.

Table 2: **Short-context comparison on DL3DV and RealEstate10K.** We report two frame budgets: N=49 (native chunk size for Voyager and SPMem; VMem uses 48) and N=121 (Gen3C native chunk). At N=49 most baselines are single-chunk; at N=121 Voyager, VMem, and SPMem all require autoregressive looping.

		N=49				N=121			
Dataset	Method	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑
DL3DV	Gen3C [24]	15.74	0.474	0.847	0.710	13.59	0.565	0.798	0.585
	Voyager [11]	12.84	0.568	0.796	0.615	11.35	0.649	0.760	0.516
	VMem [13]	11.88	0.636	0.781	0.477	10.87	0.722	0.696	0.274
	SPMem [37]	14.70	0.492	0.853	0.703	13.03	0.607	0.788	0.536
	PPMem (ours)	**16.99**	**0.335**	**0.909**	**0.861**	**13.98**	**0.464**	**0.863**	**0.769**
RE10K	Gen3C [24]	21.93	0.238	0.925	**0.949**	19.22	0.303	0.912	0.906
	Voyager [11]	17.43	0.315	0.900	0.877	15.88	0.389	0.881	0.827
	VMem [13]	11.69	0.626	0.812	0.547	11.07	0.678	0.754	0.402
	SPMem [37]	18.42	0.279	0.920	0.902	16.43	0.375	0.892	0.819
	PPMem (ours)	**22.46**	**0.168**	**0.965**	**0.949**	**19.34**	**0.254**	**0.952**	**0.910**

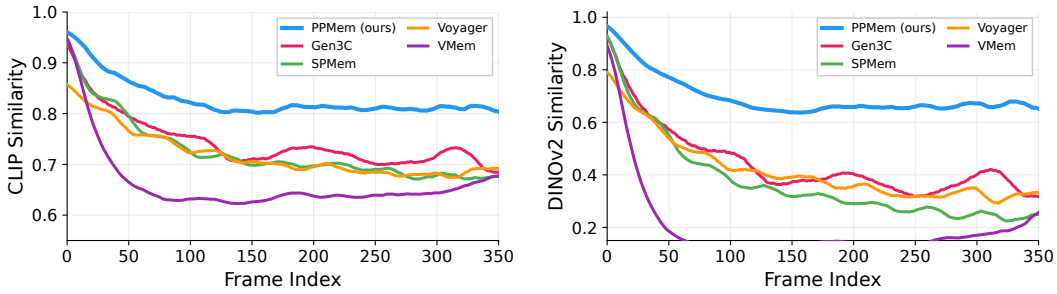


Figure 3: **Perceptual similarity vs. frame index on DL3DV long-context generation (∼350 frames).** Left: CLIP similarity. Right: DINOv2 similarity. PPMem (blue) maintains the highest perceptual fidelity across the full generation. VMem collapses early; Gen3C and SPMem degrade steadily; the gap widens with frame count, especially on DINOv2.

4.2 Short-Range Comparison

Table 2 reports short-range video quality on DL3DV and RealEstate10K. PPMem achieves the best results across all metrics on both datasets and both frame budgets, despite using a smaller base model (Wan 2.2 5B) compared to Gen3C (Cosmos 7B) and Voyager (HunyuanVideo ∼13B). Table 2 compares complete systems; Table 3 isolates the memory mechanism by swapping it on the same Wan 2.2 generator.

Beyond video quality, PPMem jointly outputs metric-scale depth. Figure 5(a) compares per-frame depth accuracy against Voyager — the only other baseline with joint depth output — over 121 frames on DL3DV. PPMem maintains lower AbsRel and higher δ<1.25 accuracy than Voyager.

4.3 Long-Context Generation under Large Viewpoint Change

We evaluate all methods on the full DL3DV test sequences (300–400 frames each), where large camera motion across many autoregressive chunks tests each method's ability to maintain coherent generation far from the initial viewpoint. Figure 3 plots CLIP and DINOv2 similarity against frame index: PPMem maintains the highest perceptual fidelity throughout, with the gap widening over time — most dramatically on DINOv2, where PPMem leads the next-best method by ∼0.27 at frame 300. Figure 4 shows a representative example: PPMem preserves scene structure and color consistency across the full sequence, while all baselines degrade visibly as the viewpoint diverges.

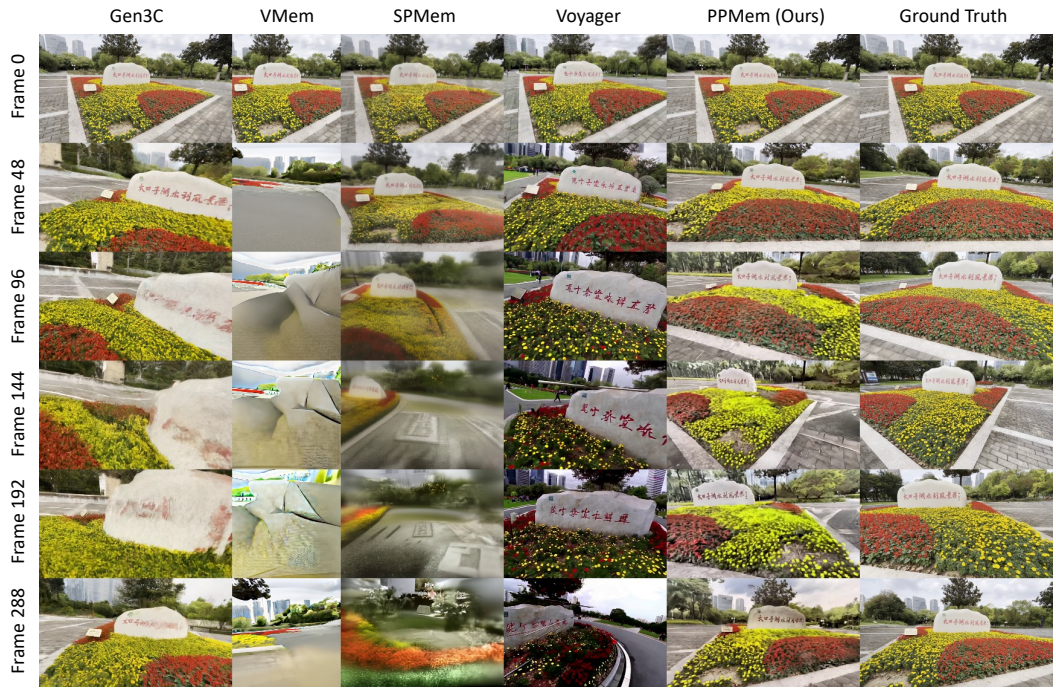


Figure 4: **Qualitative long-context comparison.** PPMem preserves scene structure and color consistency across the full sequence. VMem collapses early under large viewpoint change; SPMem grows vague and drifts; Gen3C and Voyager lose structural alignment as the viewpoint diverges.

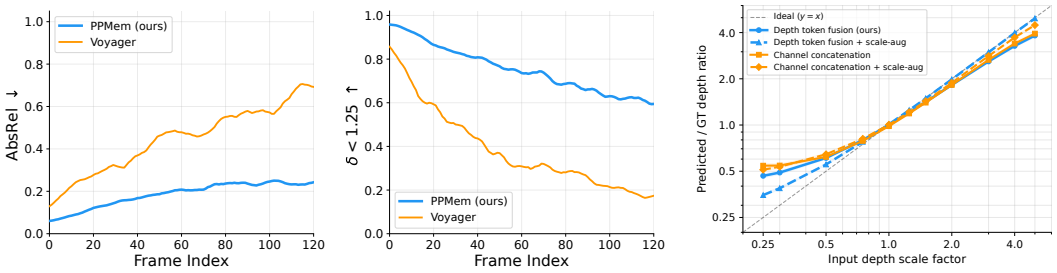
Table 3: **3D representation and prediction ablation.** We compare our joint plane pool against three inference-only alternatives (same checkpoint, no retraining): a same-model rollout without persistent memory, Gen3C-style per-frame MoGe-2 point cloud memory, and post-hoc standalone Plana3R reconstruction.

Variant	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	AbsRel↓	δ<1.25 ↑	Normal° ↓
Without persistent memory	14.58	0.407	**0.835**	0.733	0.272	64.0	25.2
MoGe-2 point cloud	14.37	0.423	0.832	0.716	0.339	55.0	25.8
Post-hoc Plana3R	14.22	0.432	0.829	0.713	0.311	57.0	26.3
Joint plane pool (ours)	**14.91**	**0.387**	**0.835**	**0.741**	**0.210**	**70.3**	**24.1**

4.4 Ablations

We ablate four design aspects below. All variants are trained on the DL3DV indoor subset at 288×512, evaluated on the 56-scene indoor test split over 3 autoregressive chunks, and share the same Stage-1 backbone and Stage-2 training recipe. Additional ablation studies on context frame composition and plane pool merging stages are in Appendix B.

3D representation and prediction (Table 3). To validate the choice of plane primitives and joint prediction, we compare three alternative configurations at inference time (no retraining): (i) **Without persistent memory**: a same-model rollout using the model's own predicted depth but no persistent memory. (ii) **Point cloud memory**: we replace the plane pool with a Gen3C-style mechanism — MoGe-2 [33] estimates per-frame monocular depth on each generated frame, and the resulting point cloud conditions the next chunk via forward warping. (iii) **Post-hoc Plana3R**: instead of using the joint plane prediction head, we run a standalone pretrained Plana3R model [18] on each generated frame post-hoc and accumulate the resulting planes into the pool. Our joint plane pool outperforms all three alternatives, with particularly large gains on depth (δ<1.25: 70.3 vs 64.0/55.0/57.0), indicating that the persistent memory contributes primarily to geometric consistency and long-horizon stability.



(a) Per-frame depth accuracy on DL3DV (121 frames). (b) DTF scale faithfulness (log-log).

Figure 5: **Depth quality analysis.** *(a)* Per-frame depth metrics (AbsRel ↓, δ<1.25 ↑) on DL3DV over 121 frames. PPMem maintains consistently lower depth error than Voyager, the only other baseline with joint depth output. *(b)* We scale the input depth conditioning by a known factor and measure the predicted/GT depth ratio. Concat + DTF (ours) tracks input scale faithfully (1.00 ± 0.01 at scale 1.0, mean $\pm$ s.e. over 56 scenes), while channel concatenation alone exhibits compressed response at extreme scales. Measured as deviation from the identity mapping over the full scale range, DTF also stays closer to it (natural-log RMSE 0.273 vs. 0.320 for concatenation alone; 0.131 vs. 0.293 with scale augmentation).

Table 4: **Depth conditioning mechanism ablation.** All variants share the same backbone and training data, varying only the depth conditioning mechanism.

Mechanism	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	AbsRel↓	δ<1.25 ↑	Normal° ↓
No depth conditioning	14.19	0.430	0.826	0.709	0.436	30.7	25.8
Channel concat only	14.88	0.388	**0.838**	**0.743**	0.212	**72.3**	**23.8**
Concat + DTF (ours)	**14.91**	**0.387**	0.835	0.741	**0.210**	70.3	24.1

Depth conditioning mechanism (Table 4). The dominant effect is the presence of depth conditioning itself: without it, depth accuracy collapses (AbsRel 0.436, δ<1.25 $= 30.7\%$) and video quality also degrades (PSNR 14.19 vs 14.91). The video improvement is indirect: better depth prediction yields a higher-quality plane pool, which in turn provides better projected conditioning (RGB, masks, rendered depth) for subsequent chunks — a cascading benefit across autoregressive steps. Given depth, channel concatenation and DTF perform comparably on the ablation benchmark, where training and evaluation both use well-aligned metric depth. At inference, the depth conditioning comes from plane pool rendering, which introduces holes, partial coverage, and scale imperfections. Figure 5b tests this via a scale faithfulness probe (whether predicted depth scales proportionally when the input depth conditioning is rescaled) — we deliberately multiply the input depth by a known factor and measure the predicted/GT depth ratio. DTF tracks input scale faithfully (1.00 ± 0.01 at scale 1.0), while channel concatenation exhibits a compressed response at extreme scales, suggesting that the VAE bottleneck discards fine-grained scale information. DTF's token-level fusion with explicit validity masking is better suited to the imperfect conditioning encountered during autoregressive inference. The two base variants are trained exclusively on well-aligned metric depth (scale ≈ 1). We additionally train both fusion schemes with per-video depth scale augmentation ($s \sim \log\text{U}[0.25, 4]$), and find that it substantially improves scale faithfulness in both cases, with DTF + scale augmentation tracking the input scale most closely overall (natural-log RMSE to the identity mapping: 0.131, vs. 0.273 for DTF alone and 0.293 for concatenation with the same augmentation).

Table 5: **Feature decoder and loss component ablations.** Top: decoder architecture (all else fixed). Bottom: loss removal (full model as baseline).

Variant	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	AbsRel↓	δ<1.25 ↑	Normal° ↓
Plana3R upsampling head	14.68	0.395	0.832	0.737	0.256	59.2	26.2
VAE-init decoder (ours)	**14.91**	**0.387**	**0.835**	**0.741**	**0.210**	**70.3**	**24.1**
– rasterization loss	14.82	0.391	0.834	0.736	0.217	66.4	25.4
– normal loss	13.27	0.502	0.807	0.675	—	—	—

Feature decoder and loss components (Table 5). Replacing the original Plana3R upsampling head [18] with a VAE-initialized decoder improves $\delta{<}1.25$ by $+11.1$ pp ($59.2 \rightarrow 70.3$) and AbsRel by -0.046 ($0.256 \rightarrow 0.210$), showing that the video VAE's learned spatial prior meaningfully benefits geometric prediction. For loss components, removing the rasterization loss degrades $\delta{<}1.25$ by -3.9 pp and Normal$^\circ$ by $+1.3^\circ$; removing the normal loss causes depth to collapse entirely (all-zero output), confirming it is critical for learning the coupled depth-normal parameterization.

5 Conclusion

We presented PPMem, a video world model that uses plane primitives as persistent 3D memory for long-context generation. The plane pool satisfies the four properties we identified for cross-chunk memory: it is compact ($\sim$100K primitives for a full scene), geometrically explicit (metric depth, normals, exportable surfaces), cross-view consistent (jointly predicted by the generation backbone), and adaptive (refined via geometric merging as generation proceeds). Experiments on DL3DV and RealEstate10K show that PPMem sustains both video quality and depth accuracy under large viewpoint changes, outperforming Gen3C, Voyager, VMem, and SPMem.

Limitations and future work. PPMem is best suited for structured scenes where planar primitives are a good geometric prior. A quantitative breakdown of performance versus scene planarity is provided in Appendix B.3. For highly organic environments (dense vegetation, water) the representation may capture less detail than point-based alternatives. Extreme camera trajectories not seen during training can also degrade quality. Our two-stage design deliberately freezes the pretrained video backbone and trains geometry through lightweight adapters, which preserves the backbone's generation quality but forgoes end-to-end co-adaptation of the video and geometry objectives; jointly optimizing both is a more general alternative we leave to future work. Extending to dynamic plane primitives and scaling to open-world scenes are natural next steps. Like all video generation models, PPMem could potentially be misused for synthetic media creation.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (62461160309) and NSFC-RGC Joint Research Scheme N_HKU705/24.

References

[1] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14834–14844, 2025.

[2] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 941–947. IEEE, 1999.

[3] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 11–20, 1996.

[4] Xinhang Gao, Junlin Guan, Shuhan Luo, Wenzhuo Li, Guanghuan Tan, et al. Memcam: Memory-augmented camera control for consistent video generation. *arXiv preprint arXiv:2603.26193*, 2026.

[5] Zelin Gao et al. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026.

[6] Google DeepMind. Genie 3: A new frontier for world models. Google DeepMind Blog, August 2025. URL `https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/`.

[7] 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.

[8] Hao He, Ceyuan Yang, Shanchuan Lin, Yinghao Xu, Meng Wei, Liangke Gui, Qi Zhao, Gordon Wetzstein, Lu Jiang, and Hongsheng Li. CameraCtrl II: Dynamic scene exploration via camera-controlled video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13416–13426, 2025.

[9] Yuze He, Wang Zhao, Shaohui Liu, Yubin Hu, Yushi Bai, Yu-Hui Wen, and Yong-Jin Liu. AlphaTablets: A generic plane representation for 3D planar reconstruction from monocular videos. *Advances in Neural Information Processing Systems*, 37:139246–139269, 2024.

[10] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[11] 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.

[12] Zeran Ke, Bin Tan, Gui-Song Xia, Yujun Shen, and Nan Xue. Interacted planes reveal 3d line mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1–17, 2026. doi: 10.1109/TPAMI.2026.3675287.

[13] Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. VMem: Consistent interactive video scene generation with surfel-indexed view memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25690–25699, 2025.

[14] Xinyang Li, Tengfei Wang, Zixiao Gu, Shengchuan Zhang, Chunchao Guo, and Liujuan Cao. FlashWorld: High-quality 3D scene generation within seconds. *arXiv preprint arXiv:2510.13678*, 2025.

[15] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

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

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

[18] Changkun Liu, Bin Tan, Zeran Ke, Shangzhan Zhang, Jiachen Liu, Ming Qian, Nan Xue, Yujun Shen, and Tristan Braud. PLANA3R: Zero-shot metric planar 3D reconstruction via feed-forward planar splatting. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 173029–173053. Curran Associates, Inc., 2025. doi: 10.52202/085713-5758. URL `https://proceedings.neurips.cc/paper_files/paper/2025/file/fc8ee7c7ab5b5f6b1615045dfb617ed6-Paper-Conference.pdf`.

[19] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. PlaneNet: Piece-wise planar reconstruction from a single RGB image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[20] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[21] Lijuan Liu, Wenfa Li, Dongbo Zhang, Shuo Wang, and Shaohui Jiao. IDCNet: Guided video diffusion for metric-consistent RGBD scene generation with precise camera control. *arXiv preprint arXiv:2508.04147*, 2025.

[22] Jinjie Mai, Wenxuan Zhu, Haozhe Liu, Bing Li, Cheng Zheng, Jürgen Schmidhuber, and Bernard Ghanem. Can video diffusion model reconstruct 4D geometry? *arXiv preprint arXiv:2503.21082*, 2025.

[23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[24] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3C: 3D-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6121–6132, 2025.

[25] Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, et al. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. *arXiv preprint arXiv:2512.14614*, 2025.

[26] Bin Tan, Nan Xue, Song Bai, Tianfu Wu, and Gui-Song Xia. PlaneTR: Structure-guided transformers for 3D plane recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4186–4195, 2021.

[27] Bin Tan, Rui Yu, Yujun Shen, and Nan Xue. PlanarSplatting: Accurate planar surface reconstruction in 3 minutes. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1190–1199, 2025.

[28] Bin Tan, Changjiang Sun, Xiage Qin, Hanat Adai, Zelin Fu, Tianxiang Zhou, Han Zhang, Yinghao Xu, Xing Zhu, Yujun Shen, et al. Masked depth modeling for spatial perception. *arXiv preprint arXiv:2601.17895*, 2026.

[29] 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.

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

[31] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025.

[32] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. MoGe: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5261–5271, 2025.

[33] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025.

[34] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20697–20709, 2024.

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

[36] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3D Gaussian splatting. *Computational Visual Media*, 10(4):613–642, 2024. doi: 10.1007/s41095-024-0436-y.

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

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

[39] Tian-Xing Xu, Xiangjun Gao, Wenbo Hu, Xiaoyu Li, Song-Hai Zhang, and Ying Shan. GeometryCrafter: Consistent geometry estimation for open-world videos with diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6632–6644, 2025.

[40] Zhen Xu, Hongyu Zhou, Sida Peng, Haotong Lin, Haoyu Guo, Jiahao Shao, Peishan Yang, Qinglin Yang, Sheng Miao, Xingyi He, Yifan Wang, Yue Wang, Ruizhen Hu, Yiyi Liao, Xiaowei Zhou, and Hujun Bao. Towards depth foundation models: Recent trends in vision-based depth estimation. *Computational Visual Media*, 12(2):243–271, 2026. doi: 10.26599/CVM.2025.9450517.

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

[42] Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. TrajectoryCrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 100–111, 2025.

[43] 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.

[44] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

A Implementation Details

A.1 Dataset Preprocessing

Training data. For ablation experiments (Tables 3–5 and Tables S2–S3), we use 5K preprocessed DL3DV [16] indoor video clips at $\mathbf{288}\times\mathbf{512}$ resolution. For the final model, we train on a mixture of DL3DV and RealEstate10K [44] (80K+ clips) at $\mathbf{576}\times\mathbf{1024}$ resolution. Each clip is paired with per-frame metric depth, surface normals, and camera intrinsics/extrinsics. Dense monocular depth and normals are estimated by MoGe-2 [33] and aligned to multi-view reconstruction from VGGT [30] via per-frame scale fitting; the median scale is applied globally to place all depths and camera translations in a consistent metric coordinate system.

Offline conditioning pipeline. A central design principle of PPMem is that *all training conditioning is generated by the same plane-based rendering process used at inference*, so that the model trains on realistic imperfections (holes, depth discontinuities, partial coverage) rather than perfect ground-truth projections. Concretely, we run an offline incremental plane reconstruction pipeline [27, 12] on each video clip to produce all conditioning signals:

1. **Incremental reconstruction.** Frames are partitioned into segments of 24. Each segment undergoes plane-based reconstruction (optimizing plane primitives to fit the aligned metric depth). After each segment, new planes are merged with the accumulated pool and jointly optimized across all frames observed so far.
2. **Conditioning generation.** After processing segment k, we render the accumulated plane pool at the *next* segment's cameras (segment $k{+}1$), producing partial depth maps that serve as the depth conditioning for that segment's training sample. This one-step look-ahead mirrors the inference-time loop, where each chunk's depth conditioning comes from rendering the pool accumulated from all *previous* chunks. Two source views are also selected from all preceding segments' frames via greedy union-coverage maximization and forward-warped to each target camera, producing projected RGB images and soft visibility masks (continuous values in $[0, 1]$ from bilinear splatting weights).
3. **Context frame selection.** Overlap frames are selected from all preceding segments via greedy union-coverage maximization over the current segment's target cameras. These are *actual video frames* (not rendered views); their accompanying depth comes from the accumulated plane pool render at those cameras.

Because the conditioning is rendered from the same plane-based reconstruction used at inference, it naturally exhibits the same artifact distribution: holes in unseen regions, depth edges at object boundaries, partial coverage from oblique viewpoints, and planar approximation errors. The gap between training and autoregressive inference thus reduces to the difference in plane quality (offline reconstruction vs. online model predictions), rather than a distribution shift between perfect ground truth and imperfect model outputs.

Context frames. For each training segment, three types of context frames are prepended along the temporal dimension:

- **Anchor**: frame 0 of the video, providing a long-range identity anchor.
- **History**: real RGB frames stratified-randomly sampled from preceding segments, each accompanied by depth rendered from the accumulated plane pool.
- **Overlap**: real RGB frames selected by greedy union-coverage maximization (as described above), with plane-pool-rendered depth.

Each context type is assigned a distinct negative RoPE temporal index, encoding it as temporally prior to the generated sequence. Context dropout is applied with probability 0.1 during training (the entire context block is dropped), encouraging robustness to partial or absent context at inference. We ablate the contribution of each context type in Table S2.

DiT input assembly. The DiT receives conditioning through three complementary pathways:

Channel concatenation. All RGB, mask, and depth signals are VAE-encoded using the Wan 2.2 separate VAE (1:1 frame mapping, $z_{\text{dim}}{=}48$). Six such latents are concatenated along the channel

dimension: the noisy video latent, two projected source-view RGB latents, two projected visibility-mask latents, and the depth conditioning latent, yielding $6 \times 48 = 288$ input channels in total.

Temporal prepend. Context frames (anchor, history, overlap) are prepended as clean tokens (timestep $= 0$) before the main frames along the temporal dimension, providing the model with real observed frames as conditioning memory.

Token-level fusion. The same depth map also enters the model at pixel resolution through the DTF module (§3.2), providing fine-grained geometric awareness via self-attention between depth tokens and DiT feature tokens.

A.2 Training Hyperparameters

Table S1: **Full training hyperparameters for Stage 2.** Stage 2 freezes the DiT backbone and injects LoRA adapters on self-attention; only geometry-specific modules and LoRA are trained. Stage 1 hyperparameters are similar except no plane loss, no DTF, no freeze/LoRA.

Category	Hyperparameter	Value
Model	Base video DiT	Wan 2.2 TI2V 5B
	VAE	Per-frame (1:1 frame mapping, z_{dim}=48)
	Depth token fusion encoder	ViT-Base ($\sim$87M params), DINOv2-Base init
	Feature decoder	Initialized from Wan VAE upsample block
	LoRA (Stage 2)	Rank 256, α=256, self-attention, all 30 blocks
	Trainable params (Stage 2)	$\sim$377M / 5.4B (7.0%)
	Precision	bfloat16
Optimization	Optimizer	AdamW
	Learning rate	1×10^{-5}
	LR warmup	50 iterations
	Total iterations (Stage 2)	5,000
	Gradient clip norm	2.0
Data	Frames per chunk	25
	Resolution	576×1024 (final) / 288×512 (ablation)
	Classifier-free guidance dropout	0.10
	Context frame dropout	0.10
Loss	λ_d (per-patch depth)	0.2
	λ_n (per-patch normal)	0.05
	λ_d^r (rasterization depth)	0.02
	λ_n^r (rasterization normal)	0.01
	λ_s (rendered depth smoothness)	0.02
	Rasterization loss warmup	1000 iterations
Depth token fusion	Mixed-patch drop probability	0.75
	Target overall mask ratio	$[0.6,\ 0.9]$

Plane head initialization. Each of the three prediction MLPs (center depth, rectangular extents, rotation quaternion) is initialized so that the last layer produces a stable constant output before training begins: depth bias is set to 2.0 m with zero weights, radii bias to 0.1 m with zero weights, and rotation bias to the identity quaternion $[1, 0, 0, 0]$ with near-zero weights (σ=0.001). During the rasterization-loss warmup (first 1000 iterations), depth and normals are supervised by per-patch losses; the extents receive no direct gradient but remain at their initialized constant, ensuring stable rasterization once the rendering loss is enabled.

A.3 Inference Setup

PPMem generates video autoregressively in 25-frame chunks; the last frame of each chunk is reused as the first frame of the next. Given an input image, a text prompt, and a target camera trajectory, the full pipeline proceeds as follows.

Initial chunk (chunk 0). The input image is used as the first frame. An initial depth estimate (from any monocular depth estimator or ground-truth depth, if available) is projected to the remaining 24

Algorithm 1 PPMem full inference loop.

Require: input image I_0, text prompt, target camera trajectory, supplied as camera-control inputs in fixed-length chunks (the last frame of each chunk seeds the next).

Initialize: plane pool $\mathcal{P} \leftarrow \emptyset$. Estimate depth for I_0 (monocular estimator, or ground-truth depth if available) and project it to chunk 0's target cameras as the initial depth conditioning (I_0 serves as both source views).
for chunk $k = 0, 1, 2, \ldots$ **do**
    Assemble conditioning: two coverage-selected source views projected to the target cameras (RGB + masks), context frames (anchor, history, overlap), and the chunk's partial depth maps — VAE-encoded for latent channel concatenation and raw for DTF.
    Video denoising: run the denoising schedule with the frozen Stage-1 backbone under classifier-free guidance $\rightarrow$ video chunk V_k.
    Geometry pass: re-run the final denoising step with the Stage-2 geometry LoRA enabled; the plane prediction head regresses plane primitives from the LoRA-adapted features; transform them to world space and rasterize them into per-frame depth/normal maps.
    Pool update and next-chunk render: merge the predicted planes into $\mathcal{P}$ (Algorithm 2); render the updated $\mathcal{P}$ at chunk $k{+}1$'s target cameras via rectangular Gaussian rasterization $\rightarrow$ partial depth maps for the next iteration.
end for
Output: generated video, per-frame depth/normal maps, final pool $\mathcal{P}$.

target cameras to produce partial RGB images and visibility masks (the single input frame is used as both source views to match the two-view conditioning format). The initial depth also serves as the depth conditioning signal for DTF. The DiT then generates the first 25-frame video clip following the same two-pass generation procedure described below (50 denoising steps with the frozen backbone, then one geometry LoRA pass); the predicted plane primitives initialize the plane pool.

Subsequent chunks (chunk $k \geq 1$). Each subsequent chunk follows a three-phase pipeline:

1. **Conditioning**: the cumulative plane pool is rendered at the next chunk's target cameras via rectangular Gaussian rasterization, producing partial depth maps. Two source views are selected from the pool's accumulated frames (greedy coverage maximization) and projected to the target cameras. Context frames — an anchor frame (frame 0 of the video), up to 5 history frames from preceding chunks, and coverage-sorted overlap frames — are prepended along the temporal dimension.
2. **Generation and plane prediction**: the DiT performs 50 denoising steps with classifier-free guidance scale 5.0, producing the generated video. We then re-run the final denoising step with the Stage 2 LoRA adapters enabled: this additional forward pass provides geometry-adapted DiT features to the plane prediction head, which regresses plane primitives. The 50-step video generation uses only the frozen Stage 1 backbone, preserving video quality; the extra LoRA step serves solely to produce plane predictions. Each predicted plane is transformed from camera space to world space using the known camera extrinsics.
3. **Pool update**: the predicted planes are merged into the cumulative pool via the merging pipeline (Algorithm 2), which removes floaters, deduplicates redundant planes, and prunes planes that will be better observed by the next chunk.

Output. After all chunks are generated, PPMem produces: (a) the full generated video, (b) per-frame metric-scale depth and surface normal maps (from the plane prediction head), and (c) the cumulative plane pool as a compact 3D reconstruction of the generated scene (exportable as a mesh via triangulation of the rectangular primitives).

Compute overhead. The plane pool adds negligible cost relative to video denoising. At 576×1024, denoising dominates each chunk ($\sim$128 s); pool rasterization ($\sim$1.7 s for 25 views) and merging ($\sim$0.3 s) are two orders of magnitude cheaper.

End-to-end memory accounting. We account separately for scene-growing, bounded per-chunk, and fixed model state. At a long-rollout snapshot, our persistent pool (106K planes, 4.8 MiB including

merger metadata) is substantially smaller than a matched point-cloud variant that accumulates per-pixel 3D points over the same rollout and resolution (1.7 GiB). Context frames, projected views, rendered depth, and merge buffers remain bounded per chunk, with a maximum simultaneous non-model state of 41 MiB, including the plane pool. DTF, the geometry decoder/head, and geometry LoRA add 1.1 GiB of fixed BF16 model state; no separate acceleration structure is used.

A.4 Plane Pool Merging Algorithm

Algorithm 2 details the full merging pipeline. Each plane carries an *observation quality score* $s = r/|\cos\theta|$ (defined in main paper §3.1), where r is the Euclidean distance to the creating camera and θ is the angle between the viewing ray and the surface normal. A lower score means the plane was observed from a closer, more frontal viewpoint and is therefore more reliable. This score drives the quality comparisons in stages 3, 5, and 6.

Algorithm 2 Plane pool merging pipeline.

Require: Pool $\mathcal{P}$, new planes $\mathcal{P}_{\text{new}}$, chunk cameras $\{(\mathbf{K}_t, \mathbf{E}_t)\}$, rendered depth $\mathbf{D}_t$, next-chunk cameras.

1: **1. Floater removal** (on $\mathcal{P}$): delete pool planes whose depth $<$ new depth $\mathbf{D}_t$ by margin δ_{occ}, if $\geq\rho$ cameras agree.
2: **2. Boundary rejection** (on $\mathcal{P}_{\text{new}}$): reject planes at depth discontinuities (Sobel gradient on patch grid $> \theta_e$).
3: **3. Within-chunk dedup** (on $\mathcal{P}_{\text{new}}$): among nearby planes ($< \epsilon_{\text{same}}$) within the chunk, keep only the one with best quality score s. Removes $\sim$80% of per-chunk planes.
4: **4. Distance filter** (on $\mathcal{P}_{\text{new}}$): reject planes farther than $d_{\max}$ from creating camera, unless pool has no coverage there.
5: **5. Cross-chunk replacement** (on $\mathcal{P} \cup \mathcal{P}_{\text{new}}$): among nearby pairs across pool and new chunk, keep the better-scored plane.
6: Append survivors to $\mathcal{P}$ with metadata.
7: **6. Forward-looking prune** (on $\mathcal{P}$): remove planes that will be viewed much closer in the next chunk ($r_{\text{next}}/r_{\text{created}} < \tau$).
8: **return** $\mathcal{P}$.

Default thresholds: δ_{occ}=0.1 (10% relative margin), θ_e=1.0 (Sobel on log-depth), ϵ_{same}=0.5 m, $d_{\max}$=5 m, ρ=0.2 (20% camera agreement), τ=0.3.

We ablate the contribution of each stage in Table S3 (Section B.2).

B Additional Ablations and Comparisons

B.1 Context Configurations

Table S2 ablates the three types of context frames that condition each autoregressive chunk: (i) *overlap*: past RGB frames selected by coverage of the target viewpoints; (ii) *history* frames from preceding chunks; and (iii) an *anchor* frame (frame 0 of the video). All configurations are evaluated at inference time without retraining, leveraging the 10% context dropout applied during training.

Component ablation (top). Removing all context degrades δ<1.25 from 70.3 to 64.2. Overlap frames alone recover most of this gap (+5.0 pp) and achieve the lowest AbsRel (0.208). Adding history and anchor further improves PSNR and perceptual metrics, with the full combination achieving the best balance across all metrics.

History scaling (bottom). Increasing the number of history frames beyond the default (5) yields only marginal gains (PSNR 14.91 $\rightarrow$ 14.94) while prefix sequence length and attention cost grow with the history count. We keep 5 as the default for this quality/compute tradeoff.

Table S2: **Context frame configuration ablation, inference-only.** We vary the context frame composition at inference time without retraining (the model was trained with context dropout 0.1). Anchor = frame 0 of the video; History = frames from preceding chunks; Overlap = past RGB frames selected by coverage of the target viewpoints. Top: component ablation. Bottom: history count scaling. Evaluated on 56 DL3DV indoor scenes at 288×512.

	Anchor	History	Overlap	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	AbsRel↓	δ<1.25 ↑	Normal° ↓
Component ablation										
No context	✗	✗	✗	14.60	0.403	**0.835**	0.730	0.219	64.2	24.5
Overlap only	✗	✗	✓	14.73	0.394	0.834	0.738	**0.208**	69.2	24.3
History + Overlap	✗	✓	✓	14.77	0.393	0.832	0.739	0.218	70.2	24.3
All (default)	✓	✓	✓	14.91	**0.387**	**0.835**	**0.741**	0.210	**70.3**	24.1
History scaling (Anchor + Overlap + varying history count)										
Hist = 5 (default)	✓	✓	✓	14.91	**0.387**	**0.835**	**0.741**	0.210	**70.3**	24.1
Hist = 10	✓	✓	✓	14.80	0.393	0.833	0.735	0.220	69.0	24.2
Hist = 20	✓	✓	✓	14.82	0.393	0.834	0.733	0.225	68.6	24.2
Hist = 36	✓	✓	✓	**14.94**	0.390	0.833	0.732	0.227	67.9	**24.0**

B.2 Plane Pool Merging Phase Ablation

Table S3 validates each merging mechanism by starting from the baseline (no merging improvements) and progressively adding one at a time. Each row includes all mechanisms from the rows above it. Within-chunk dedup (stage 3) provides the largest single improvement, reducing per-chunk plane count by $\sim$4.5$\times$ (518K $\rightarrow$ 116K) while improving δ<1.25 on both indoor and outdoor scenes. Every subsequent addition further improves δ<1.25 monotonically on both splits. Boundary rejection uses a single global threshold shared across indoor and outdoor scenes (no per-domain tuning), trading a small amount of coverage for stricter geometry filtering.

Table S3: **Progressive build-up of the plane merging pipeline.** Starting from the baseline, we add each mechanism one at a time; each row includes all mechanisms from the rows above it. Depth accuracy is measured by absolute relative error (Abs Rel, lower is better) and the percentage of pixels with predicted/GT depth ratio within 1.25 (δ<1.25, higher is better). Coverage is the fraction of pixels receiving a valid rendered depth. Evaluated on 16 DL3DV scenes (8 outdoor + 8 indoor).

	Outdoor (8 scenes)			Indoor (8 scenes)			
Mechanism (cumulative)	Abs Rel↓	δ<1.25 ↑	Coverage	Abs Rel↓	δ<1.25 ↑	Coverage	Planes
Baseline (no merging improvements)	0.479	41.1	0.865	0.311	46.6	0.948	518K
+ Within-chunk dedup (stage 3)	0.486	44.7	0.833	0.297	51.8	0.920	116K
+ Cross-chunk replacement (stage 5)	0.477	47.0	0.803	0.296	52.7	0.916	100K
+ Forward-looking prune (stage 6)	**0.469**	48.3	0.798	0.285	53.9	0.908	100K
+ Boundary rejection (stage 2)	0.472	**49.6**	0.786	**0.284**	**56.5**	0.900	**97K**

Accumulated plane pool. We directly evaluate the accumulated plane pool by rendering each final pool over the complete generated trajectory. Across 56 scenes, the accumulated pool compresses the per-chunk plane predictions into only $\sim$100K planes while retaining accuracy close to them (0.241 vs. 0.210 AbsRel), with 98.7% image coverage. This directly measures both geometric fidelity and completeness. Figure S1 compares the pool against Plana3R reconstructions of the same generated videos.

B.3 Scene Planarity

We compute scene planarity independently of PPMem predictions. For every view in the reference sequence, we use estimated depth and surface normals to group neighboring valid pixels satisfying surface-normal and local 3D plane-consistency constraints. Scene planarity is the fraction of valid-depth pixels, pooled across views, covered by connected planar regions of at least 256 pixels. Across the 100 DL3DV scenes, planar coverage has a Spearman correlation of 0.33 with PSNR at 121 frames. Table S4 groups the scenes into within-domain tertiles of planar coverage.

Overall, lower planarity is associated with a gradual decrease in PSNR and depth accuracy, while LPIPS varies only mildly across the three groups.

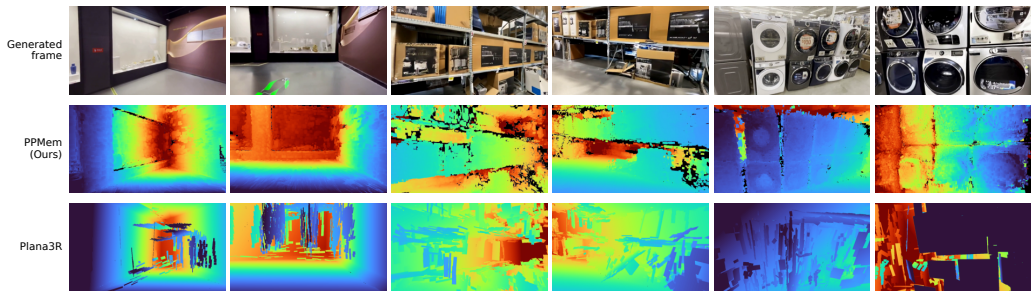


Figure S1: **Accumulated plane pool vs. Plana3R reconstruction.** We render depth from our plane pool and from planes reconstructed by Plana3R [18] on the same generated video. The pool built online during generation stays consistent, whereas pairwise reconstructions conflict across views and fragment.

Table S4: **Performance by scene planarity.** The 100 DL3DV scenes are grouped into within-domain tertiles of planar coverage; all metrics are computed at $N{=}121$.

Planar coverage	Low	Middle	High
# Scenes	34	32	34
PSNR↑	12.98	13.67	15.27
LPIPS↓	0.482	0.466	0.443
AbsRel↓	0.289	0.250	0.195
$\delta{<}1.25$ ↑	64.2	72.5	79.0

B.4 Comparison to Non-3D-Persistent Baselines

We additionally evaluate WorldPlay [25] and MemCam [4], which use historical-frame context without an explicit persistent 3D representation, on the same DL3DV evaluation set (Table S5).

Table S5: **Comparison to non-3D-persistent baselines on DL3DV.** WorldPlay and MemCam condition on historical frames without an explicit persistent 3D representation. The evaluation set and frame budgets match Table 2.

	$N{=}49$				$N{=}121$			
Method	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑	PSNR↑	LPIPS↓	CLIP_{sim} ↑	$\text{DINOv2}_{\text{sim}}$ ↑
WorldPlay [25]	13.22	0.532	0.883	0.784	11.98	0.604	0.848	0.683
MemCam [4]	12.35	0.592	0.782	0.614	11.08	0.657	0.732	0.472
PPMem (ours)	**16.99**	**0.335**	**0.909**	**0.861**	**13.98**	**0.464**	**0.863**	**0.769**

C Additional Qualitative Results

Figures S3 and S4 show two additional long-context comparisons on DL3DV scenes (outdoor rooftop and indoor store), confirming the trends from the main paper. Figure S2 shows a representative failure case where fine text degrades under zoom-in.



Figure S2: **Failure case: fine text rendering under zoom-in.** Scene structure and prominently visible text are preserved across 288 frames, but text that is barely legible in the initial viewpoint becomes garbled as the camera approaches. Faithful text synthesis from low-resolution observations remains an open challenge for video diffusion models.

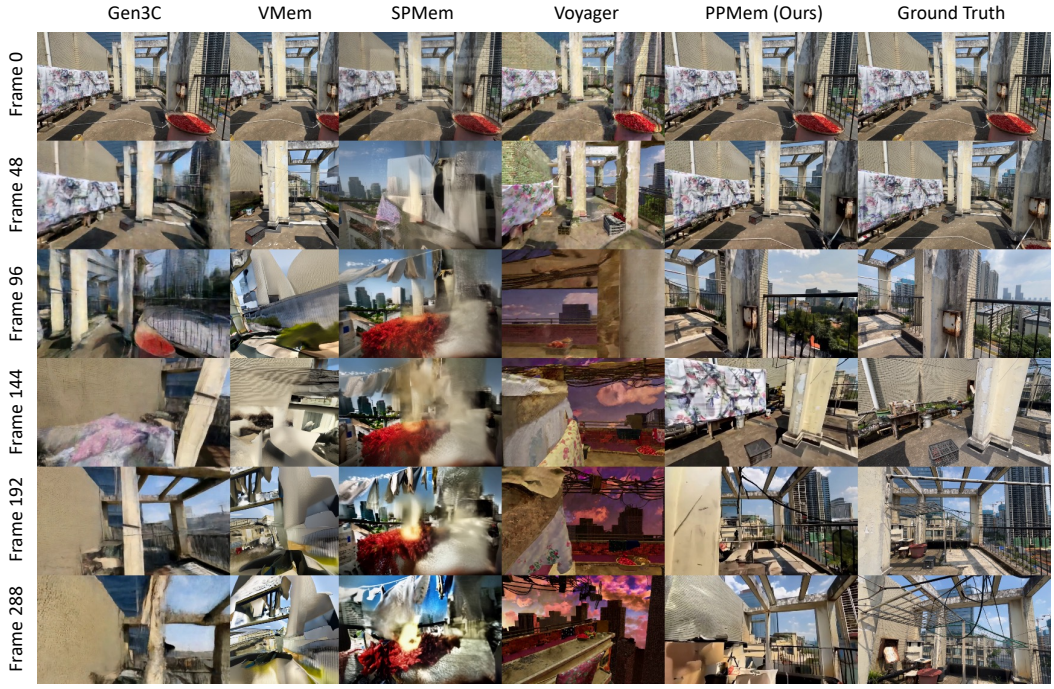


Figure S3: **Additional qualitative comparison: outdoor rooftop scene.** PPMem maintains structural consistency across 288 frames under large viewpoint change.



Figure S4: **Additional qualitative comparison: indoor store scene.** PPMem preserves scene layout and object identity while all baselines degrade.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Contributions and scope are stated in the abstract and §1. Limitations are discussed in §5.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: §5 discusses the static-scene assumption, extreme trajectory degradation, and planar prior scope.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: No theoretical results or proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Architecture in §3, training/eval setup in §4.1, full hyperparameters in Appendix Table S1, inference pipeline in Appendix A.3, merging algorithm in Appendix A.4.

   Guidelines:

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

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

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

Answer: [No]

Justification: All datasets used are publicly available. Inference code, pretrained model weights, and usage instructions are publicly available at `https://github.com/robbyant-research/PPMem`.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: §4.1 specifies data, splits, baselines, resolution, and metrics. Appendix Table S1 provides the full hyperparameter table.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: We report means over 200 test scenes. Per-run error bars are omitted due to the high compute cost of video generation. The DTF scale test (Figure 5b) reports mean $\pm$ s.e. across 56 scenes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Training uses 32× A800 GPUs with bfloat16 precision. Iteration counts are reported in Appendix Table S1.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available datasets and models and does not involve human subjects.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: §5 notes the potential for misuse in synthetic media generation.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: No models or datasets with high misuse risk are released.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All assets are cited. DL3DV: CC-BY-NC 4.0; RealEstate10K: CC-BY 4.0; Wan 2.2: Apache 2.0; DINOv2: Apache 2.0; MoGe: MIT; Plana3R: MIT. All used for non-commercial research.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We release inference code and pretrained model weights with installation and usage instructions in the code repository and architecture and license information in the model card, linked from `https://hyzcluster.github.io/ppmem`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not a component of the core method.

    Guidelines:

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