RETHINKING LARGE VIEW SYNTHESIS MODELS: CAPACITY AND SCENE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large view synthesis models (LVSMs) employ diverse architectures and scene representations, but how to allocate capacity and computation for effective scaling remains unclear. We organize these systems through scene memory constructed from context views and queried for novel view synthesis. We systematically study network organization, spatial granularity, model capacity, and the allocation of resources between scene memory and decoding. Within the evaluated full-attention configurations, decoder-only models favor reconstruction quality, whereas separate decoders favor rendering efficiency. Spatial granularity is a key scaling dimension alongside model capacity, and the benefits of larger scene memory depend on its decoder. These findings lead us to select and train a ViT-L/16 backbone, yielding highly competitive reconstruction quality with shorter reported reconstruction times than most baselines on DL3DV. Using this backbone, we analyze sparse attention for novel view synthesis. Training-free block-sparse attention preserves spatial tokens and scene-memory capacity, accelerating scene construction by 1.51–$2.22\times$ over dense inference at fixed inputs with only a small quality loss. Further analysis shows how layer sensitivity and routing overhead govern the quality and speed gains. We hope our study helps the community scale up LVSMs more effectively and explore better model designs.

1 INTRODUCTION

Large view synthesis models (LVSMs) perform novel view synthesis (NVS) from context images, offering a feed-forward route to scene reconstruction. Recent systems pursue this goal through diverse designs, including Gaussian representations with splatting (Zhang et al., 2025c; Chen et al., 2025; 2026), attention-based scene representations (Jin et al., 2025; Sun et al., 2026), and scene-specific fast weights (Zhang et al., 2026; Wang et al., 2026a). These designs offer several ways to invest resources: enlarge the shared network, retain finer spatial features, increase scene-specific storage, or strengthen target-view decoding. Understanding which investments improve reconstruction, and at what cost, is essential to scaling these models effectively.

These investments are difficult to disentangle when evaluated only as parts of complete systems. Parameter count does not capture token-processing cost, and a compact representation can still require expensive decoding. We use a common functional view to organize these choices: context images construct *scene memory*, which target-camera queries access to synthesize novel views (Figure 1). Separating shared parameters, scene storage, and construction and rendering costs leads to three questions.

Q1. How should reconstruction networks be organized and scaled? We examine shared and separate modules for scene construction and target decoding, and compare parameter scaling with finer spatial tokenization. Within the evaluated full-attention configurations, decoder-only models favor quality and separate decoders favor rendering throughput. Finer tokens improve quality at substantial computation and storage costs, identifying a scaling dimension beyond model size.

Q2. How should capacity be allocated between scene memory and its decoder? We study key–value (KV) and Gaussian representations together with their query mechanisms. Their relative benefits depend on both storage and decoding capacity: a larger memory does not consistently improve quality, and increasing query depth can impose a substantial rendering cost. These observations motivate evaluating scene memory and its decoder jointly.

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.

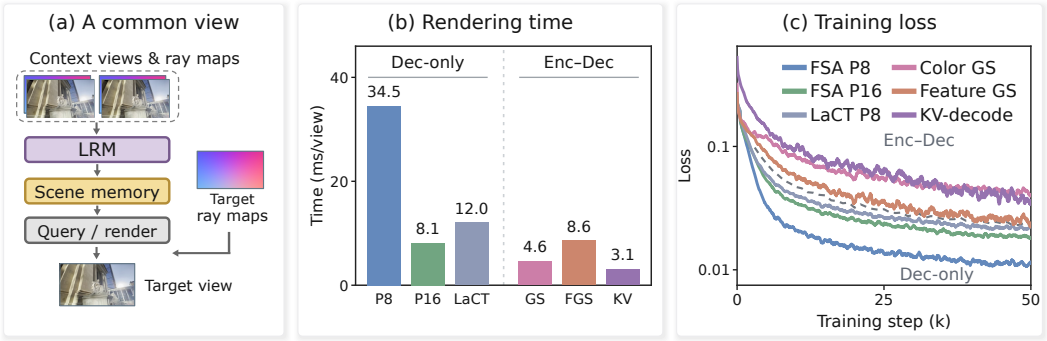

Figure 1: Six representative systems (FSA: full self-attention; FSA P16: 306M; FeatureGS: F32/D1024). (a) A common construction–query view illustrated with RGB images and ray-direction maps; the output thumbnail is a reference target image. (b) Rendering time ($1000/\mathrm{FPS}$). (c) Reconstructed training loss with a trailing 500-step mean; the dashed guide separates architectural groups. Colors match across (b,c); regularization and logging differ across families.

Q3. Can scene construction be accelerated while preserving spatial tokens and scene-memory capacity? The preceding comparisons motivate a decoder-only ViT-L/16 backbone as a practical trade-off between reconstruction quality and rendering throughput. We train this configuration at high resolution and use it to study sparse attention. As cross-view interactions increasingly dominate construction cost, we examine block selection while retaining spatial tokens and scene-memory capacity. The analysis reveals how layer sensitivity and routing overhead determine the achievable speedup and quality loss.

Our study aims to help the community direct scaling resources toward the components that most effectively improve view synthesis. We make three contributions:

- **An empirical study of capacity and scene memory.** We compare network organization, spatial granularity, scene representations, and decoder capacity to identify their effects on reconstruction quality and cost.
- **An exploration of sparse attention for novel view synthesis.** We investigate sparse cross-view computation, revealing how layer sensitivity and routing overhead affect reconstruction quality and acceleration.
- **A model with strong reconstruction quality and fast scene construction.** Building on this empirical study, we train a ViT-L/16 backbone at high resolution and augment it with sparse attention to obtain the Block-Sparse View Synthesis Model (BVSM).

2 Capacity, Scene Memory, and Study Design

Scene construction and querying. Given posed context images $\mathcal{O} = \{(I_v, c_v)\}_{v=1}^{V_c}$, a reconstruction system constructs scene memory S and queries it with a target camera c_t:

$$S = E_\theta(\mathcal{O}), \qquad \widehat{I}_t = R_\phi(S, c_t). \tag{1}$$

This functional decomposition allows E_θ and R_ϕ to share weights. Scene memory is the scene-specific information retained across target queries, distinct from shared model parameters. KV caches, fast weights, and Gaussians instantiate this role with different query mechanisms (Figure 1).

Architectural and capacity axes. Here, *decoder-only* denotes shared context-processing and target-prediction weights, without implying autoregressive generation. *Encoder–decoder* denotes a separate query stage: Color GS rasterizes directly to RGB, FeatureGS adds learned feature decoding, and KV decoders query stored keys and values. Independently, FSA and LaCT distinguish full-attention context fusion from nonlinear fast weights (Zhang et al., 2026). Network width and depth control model capacity; patch size controls spatial granularity; scene-state dimensions and decoder width or depth determine storage and querying capacity.

Table 1: Architecture and capacity comparisons at 144×256. Tokens are per view; scene memory is for 32 context views. F/D: feature/decoder width; r: cross-attention depth. Best PSNR/FPS among displayed rows in each group are bold.

ID	Patch / tokens	Network architecture	Scene state	Total params.	Scene mem. (MiB)	Train TFLOPs↓	PSNR↑	Render FPS↑
A. Spatial granularity and model capacity								
Patch: $1 \leftrightarrow 2$ Params: $2 \rightarrow 3 \rightarrow 4$								
1	8 / 576	FSA · Dec-only	KV cache	171M	648.0	94.01	**26.44**	29
2	16 / 144	FSA · Dec-only	KV cache	173M	162.0	9.47	23.79	**197**
3	16 / 144	FSA · Dec-only	KV cache	306M	216.0	14.72	24.18	123
4	16 / 144	FSA · Dec-only	KV cache	608M	432.0	29.33	24.60	62
B. Network architecture and scene state								
Reference: 1								
1	8 / 576	FSA · Dec-only	KV cache	171M	648.0	94.01	**26.44**	29
5	8 / 576	LaCT · Dec-only	Fast weight	157M	81.0	54.88	23.12	83
			↑ Dec-only Enc–Dec ↓					
6	8 / 576	FSA · Enc–Dec	Color GS	173M	132.8	94.00	21.13	217
7	8 / 576	FSA · Enc–Dec	Feature GS (64)	255M	168.8	94.68	22.88	103
8	8 / 576	FSA · Enc–Dec	KV ($r = 4$)	264M	216.0	84.72	23.25	**325**
C. FeatureGS memory and decoder capacity								
Features: $9 \leftrightarrow 10$ Decoder width: $10 \leftrightarrow 7$ 11: two fewer MLPs								
9	8 / 576	FSA · Enc–Dec	F32 / D768	217M	96.8	94.39	18.74	**571**
10	8 / 576	FSA · Enc–Dec	F64 / D768	221M	168.8	94.57	22.24	154
7	8 / 576	FSA · Enc–Dec	F64 / D1024	255M	168.8	94.68	22.88	103
11	8 / 576	FSA · Enc–Dec	F32 / D1024	234M	96.8	94.44	**23.00**	116
D. KV memory and cross-attention depth								
Memory: $13 \leftrightarrow 8$, $14 \leftrightarrow 16$ Depth: $8 \leftrightarrow 14$								
13	8 / 576	FSA · Enc–Dec	KV ($r = 4$)	245M	54.0	82.63	23.01	**337**
8	8 / 576	FSA · Enc–Dec	KV ($r = 4$)	264M	216.0	84.72	**23.25**	325
14	8 / 576	FSA · Enc–Dec	KV ($r = 6$)	280M	216.0	94.93	23.18	246
16	8 / 576	FSA · Enc–Dec	KV ($r = 6$)	299M	324.0	97.01	22.60	240

Study design. Table 1 compares network organization and scene state (B), spatial granularity and model scale (A), and memory/decoder allocation in FeatureGS (C) and KV systems (D). Configurations are trained separately at 144×256, with 32 context and 32 target views, global batch 32, and a 50K-step budget. The objective is MSE plus $0.2\times$LPIPS (Zhang et al., 2018), with opacity regularization for Gaussian variants. Equal training steps do not imply equal computation. These comparisons guide backbone selection; subsequent sparse-attention analysis holds the trained weights and inputs fixed.

Resource accounting. Total parameters include construction and learned query modules, counting shared weights once. Scene memory measures persistent payload for 32 context views, excluding shared parameters, transient activations, and workspaces; KV and Gaussian sizes use a common BF16 storage convention. Estimated training TFLOPs are per scene sample and exclude Gaussian splatting. PSNR measures reconstruction quality; construction time measures building S, while rendering FPS measures target-query throughput. Persistent payload differs from peak runtime memory: sparse interactions can reduce computation while retaining scene-memory capacity.

3 Understanding Design Choices in LVSMs

We first examine the organization of construction and querying, then study scaling within decoder-only and encoder–decoder systems. Table 1 supports four findings on these design choices; Figure 1(c) complements the evaluation with training diagnostics. Together, the comparisons guide the backbone choice for the high-resolution study.

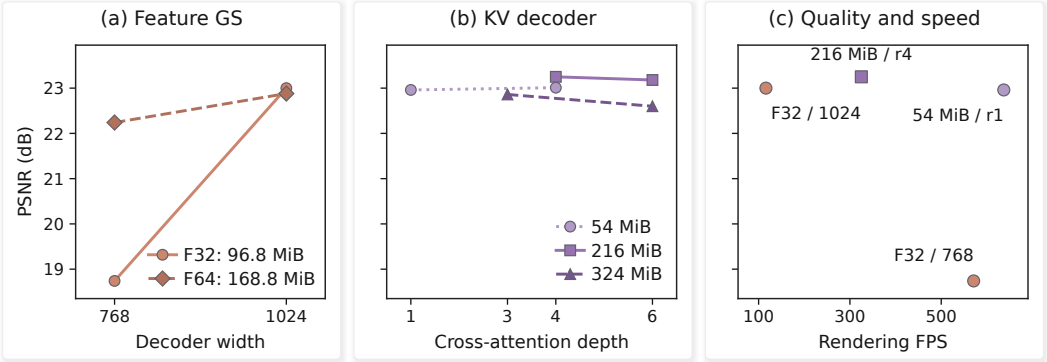

Figure 2: Capacity allocation in encoder–decoder models. (a,b) Evaluated configurations at fixed BF16 scene memory per curve. F32/D1024 has six decoder MLPs; the other FeatureGS runs have eight. KV refinement remains fixed. (c) Highest-PSNR and highest-FPS configurations within each family. Labels give feature/decoder width or KV memory/query depth r. Lines connect measurements; the F32 curve varies both decoder width and depth.

3.1 Network Organization

The first choice is whether context processing and target prediction should share a backbone.

Finding 1. Within the evaluated FSA configurations, decoder-only reconstruction achieves the highest quality, while separate encoding and decoding support faster rendering.

Table 1B–C exposes a trade-off between image quality and repeated-query cost. With fixed query batching, total latency for Q target views is approximately $T(Q) = T_{\text{build}} + QT_{\text{query}}$. A cheaper query stage becomes increasingly valuable as more views are rendered from the same scene. The preferred organization therefore depends on both quality requirements and the intended query workload.

Memory design and decoder capacity also change in these system comparisons. LaCT further changes the context-processing mechanism; its lower quality, computation, and storage reflect a distinct design trade-off.

3.2 Scaling Decoder-Only Models

Within decoder-only models, capacity and spatial granularity offer different routes to improving quality.

Finding 2. Spatial granularity is a key scaling dimension alongside model capacity, with substantial implications for reconstruction quality and computational cost.

At fixed patch size, increasing capacity improves quality. Yet the largest P16 model remains below the smaller P8 reference (Table 1A). Finer tokens yield a larger quality gain in this comparison, accompanied by higher computation, scene memory, and rendering cost.

For V_c views of size $H \times W$ and patch size P, $N_c = V_c HW/P^2$ context tokens require $\Theta(LN_cD)$ KV storage across L stages of width D, and $\Theta(N_c^2 D)$ global-attention computation per layer. Thus patch size changes both the stored representation and the cost of accessing it. Parameter count alone cannot capture this scaling cost. The patch intervention affects both context and target tokens; equal-step training does not establish a compute-optimal configuration.

3.3 Scene Representations for Encoder–Decoder Models

With a separate query stage, the choice concerns a representation–decoder pair, (S, R_ϕ).

Finding 3. Among the evaluated encoder–decoder systems, the reference KV decoder offers a favorable balance of reconstruction quality and rendering speed, while FeatureGS uses less scene memory.

The KV reference achieves higher quality and throughput than Color GS and most FeatureGS variants, despite storing more scene memory (Table 1B–C). The fastest FeatureGS setting instead combines compact storage with high throughput at a substantial quality cost. These operating points show why storage alone cannot predict rendering efficiency: the work required to query the representation is also consequential. The comparison evaluates complete systems, with representation, decoder capacity, and optimization jointly affecting the outcome.

3.4 Allocating Capacity between Memory and Decoder

We next vary memory and decoder capacity within each representation family (Table 1C–D).

Finding 4. Scene-memory size alone is insufficient to predict reconstruction quality or rendering cost; capacity must be allocated jointly to the representation and its decoder.

For FeatureGS, increasing feature width at $D = 768$ and widening the F64 decoder both improve quality at fixed refinement depth. F32/D1024 attains the highest measured FeatureGS PSNR with compact memory, but uses six MLPs versus eight in the other runs. Comparisons involving it therefore couple width and depth (Figure 2(a)).

For KV, deeper querying imposes a larger throughput penalty than the evaluated memory increases, without consistently improving quality. More retained memory also does not consistently improve quality (Figure 2(b)). The compact, one-query-layer configuration illustrates an alternative operating point with faster rendering and a small quality concession (Figure 2(c)). These are separately trained systems; changing retained stages also changes content and projection parameters, and single-seed differences do not establish significance.

Implication for backbone selection. We prioritize decoder-only quality, and choose the 306M P16 configuration in Table 1A as a practical compromise: it improves quality over the 173M P16 model, uses fewer tokens than P8, and renders faster than the 608M configuration. Section 4 evaluates this choice after high-resolution training and studies sparse computation within the selected backbone.

4 Block-Sparse View Synthesis Models

The preceding findings motivate a 306M P16 decoder-only backbone, which we train at high resolution. With this backbone fixed, we study how sparse cross-view computation can accelerate scene construction while retaining spatial tokens and scene-memory capacity. The resulting system is the *Block-Sparse View Synthesis Model* (BVSM).

4.1 Full-Attention Backbone

BVSM-Dense (Dense) alternates within-view and cross-view attention (Figure 3). Following DVSM (Sun et al., 2026), context processing stores layerwise KV memory for target views to query through the same backbone. Construction and rendering thus share weights without a separate decoder.

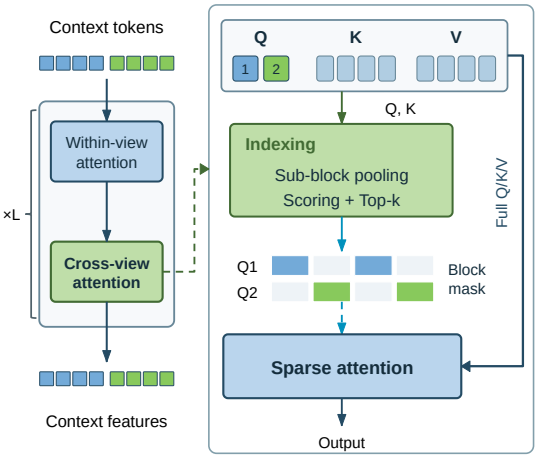

Figure 3: Sparse attention in selected context cross-view layers. All tokens are retained; target rendering remains dense.

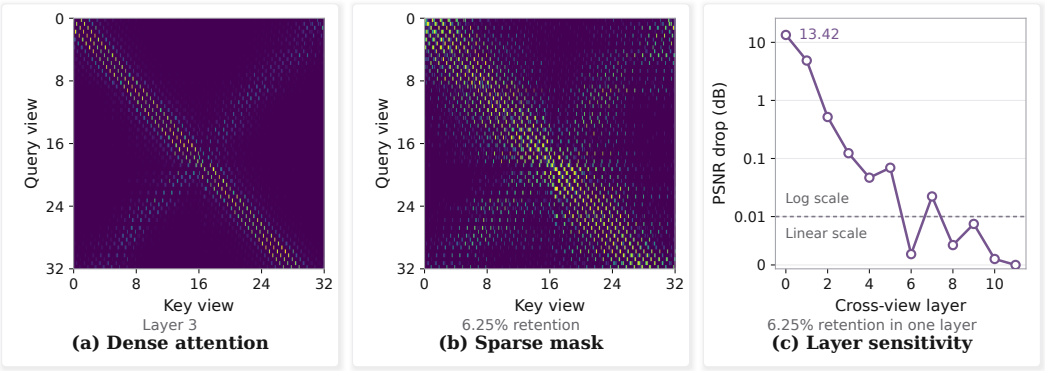

Figure 4: Attention structure and layer sensitivity. (a,b) Layer-3 dense attention and its sparse mask at 6.25% retention. (c) PSNR drop when only the indicated cross-view layer is sparsified to 6.25% retention. This diagnostic uses six scenes, 32 context views, and 16 targets per scene, with a dense baseline of 29.143 dB. The vertical scale is logarithmic above 0.01 dB and linear below, preserving zero.

For one head, attention takes the standard form

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V, \tag{2}$$

where d is the head dimension. Within-view attention operates independently on each image; cross-view attention accesses all context keys. During scene construction, V_c views with n tokens each and hidden width D incur cross-view interaction cost $\Theta(V_c^2 n^2 D)$. This quadratic growth motivates sparsifying construction while keeping target rendering dense.

4.2 Block-Sparse Scene Construction

BVSM-Sparse, or Sparse, replaces selected cross-view operations with input-dependent block selection, without updating network weights. For each layer and head, *indexing* selects key blocks for every query block; *sparse execution* evaluates attention over the selected interactions.

Indexing. We summarize each block by normalized means of its sub-blocks, denoted $\widehat{q}_{i,a}$ and $\widehat{k}_{j,b}$. Their similarities determine the score of query block i and key block j:

$$s_{ij} = \log \sum_{a,b} \exp\left(\widehat{q}_{i,a}^\top \widehat{k}_{j,b}\right), \qquad \mathcal{J}_i^{\ell,h} = \mathrm{TopKIndices}_j(s_{ij}, k_\ell). \tag{3}$$

Here a, b index sub-blocks, and k_ℓ is the number of retained key blocks determined by the layer's retention budget ρ_ℓ. Sub-block descriptors summarize variation within each block; log-sum-exp emphasizes stronger matches. Selection is independent for each query block and head.

Mask and execution. The indices $\mathcal{J}_i^{\ell,h}$ define a block mask (Figure 4b). Its token-level counterpart $M^{\ell,h}$ is zero for selected block pairs and $-\infty$ otherwise, giving

$$O^{\ell,h} = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M^{\ell,h}\right) V. \tag{4}$$

We execute this masked operation using selected-block indices and SpargeAttention's quantized sparse backend (Zhang et al., 2025b); reported results therefore include both block selection and quantized execution. All context tokens remain in the feature stream, preserving KV layout and capacity although cached values can change. The budget ρ_ℓ controls retained interactions, while target rendering still uses full attention. Architecture and implementation details are given in Appendix B.

4.3 Analysis

We evaluate the high-resolution backbone on DL3DV (Ling et al., 2024) at 544×960, following the evaluation protocol of Long-LRM (Chen et al., 2025), with 133 valid scenes and 5,901 target views.

Table 2: Multi-view reconstruction quality and efficiency on DL3DV. BVSM-Dense and BVSM-Sparse are evaluated with 16, 32, and 64 input views at 544×960. Runtime values retain their source measurement conditions.

Method	16 input views				32 input views				64 input views			
	PSNR ↑	SSIM ↑	LPIPS ↓	Recon. (s)↓	PSNR ↑	SSIM ↑	LPIPS ↓	Recon. (s)↓	PSNR ↑	SSIM ↑	LPIPS ↓	Recon. (s)↓
3DGS	21.20	0.708	0.264	> 600	23.60	0.779	0.213	> 600	26.43	0.854	0.167	> 600
LVSM	21.64	0.666	0.365	4.00	21.73	0.664	0.365	15.00	21.46	0.651	0.377	60.00
Long-LRM	22.66	0.740	0.292	0.40	24.10	0.783	0.254	1.00	24.77	0.804	0.239	3.00
Long-LRM++	24.40	0.795	0.231	1.60	26.43	0.846	0.180	4.70	27.30	0.869	0.161	16.00
tttLRM	23.60	0.784	0.255	3.60	25.07	0.822	0.215	7.20	25.95	0.844	0.195	15.00
LaCT	24.70	0.793	0.224	15.00	26.90	0.837	0.185	29.00	28.30	0.857	0.169	59.00
BVSM-Dense	**25.22**	**0.804**	**0.166**	0.51	**27.42**	**0.857**	**0.126**	1.56	**29.28**	**0.891**	**0.102**	5.41
BVSM-Sparse	24.98	0.798	0.171	**0.33**	27.22	0.853	0.129	**0.84**	29.13	0.889	0.103	**2.43**

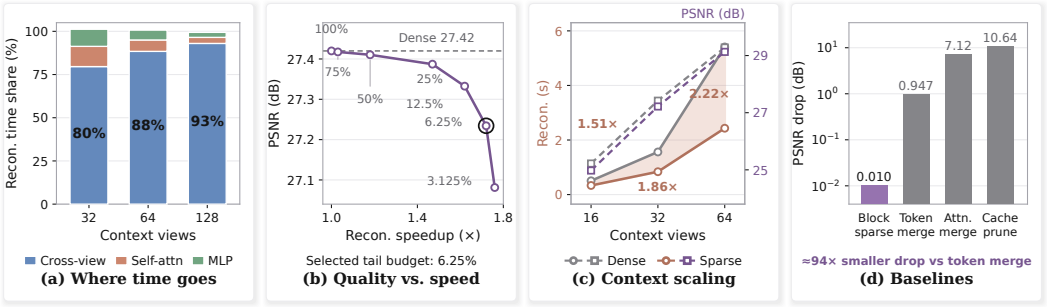

Figure 5: Sparse reconstruction. (a) Independent module timings from an earlier checkpoint; shares are not renormalized. (b) Retention sweep over layers 4–11 at 32 views, with layers 0–3 dense; the circle selects the tail budget. (c) Dense versus Sparse: circles denote time, squares denote PSNR. Quality uses 133 scenes, latency a single scene. (d) Training-free baselines at 32 views. Block sparsity and token merging share layers 6–11 and 25% nominal retention; the other baselines use different settings.

Dense and Sparse use identical inputs and the same checkpoint. Reconstruction and rendering are timed separately on a single scene with an A100. The layer-sensitivity diagnostic uses six scenes; the module profile in Figure 5(a) uses an earlier checkpoint of the same architecture.

A strong full-attention reference. Dense achieves the best reconstruction quality among the methods listed in Table 2, with shorter reported construction times than most baselines. It provides a strong reference for studying the cost of approximating cross-view interactions. Quality protocols are aligned across methods, while reference runtimes retain their source conditions; our acceleration claims use paired Dense/Sparse measurements.

Finding 5. At fixed inputs and weights, the evaluated block-sparse implementation accelerates scene construction with a small loss of image quality while preserving spatial tokens and scene-memory capacity.

Quality and construction time. Sparse accelerates construction by 1.51–$2.22\times$, with larger gains at longer context and a small quality gap (Figure 5(c)). This trend is consistent with the growing share of cross-view computation in Figure 5(a). Rendering latency and reported peak memory remain essentially unchanged, separating the benefit of reducing interactions from that of reducing scene storage or query cost.

Sparse interactions versus token merging. At the same intervened layers and nominal retention, block sparsity incurs substantially less quality loss than token merging (Figure 5(d)). This favors



Figure 6: Qualitative comparison of BVSM-Dense, BVSM-Sparse at two retention settings, and 6.25% retention in all context cross-view layers. Each panel shows the reconstruction (left) and error relative to ground truth (right), separated by a slanted line. Inset PSNR values correspond to the displayed example.

retaining spatial tokens while approximating their interactions in the tested setting. Equal retention does not imply equal computation; attention merging and cache pruning also use different settings. The comparison identifies a promising intervention without establishing superiority at matched system cost.

Finding 6. In the evaluated backbone, early cross-view layers are more sensitive to sparsification, whereas later layers tolerate aggressive sparsity with a small quality loss.

Sensitivity across depth. The attention map and mask illustrate block selection (Figure 4(a,b)), but its effect on the final image depends on where it is applied. At 6.25% retention, single-layer interventions cause much larger quality losses in the first two layers than in layers 4–10 (Figure 4(c)). The qualitative examples in Figure 6 show the consequence of aggressive and all-layer sparsification. These observations favor protecting early computation. The terminal zero requires separate interpretation: when no later state-producing module uses its output, the final attention update cannot affect rendering through the cache.

Retention and end-to-end speed. The tail-layer sweep shows that later layers can also be sparsified jointly with a small quality loss (Figure 5(b)). At the lowest retention levels, further block removal yields little additional speedup while quality deteriorates. Routing, projections, within-view attention, MLPs, and dense layers remain part of construction cost, so retention should be chosen from measured quality and latency. This sweep holds the first four layers dense and differs from the complete Sparse configuration in panel (c).

Routing quality and overhead. At the same block budget and sparse backend, sub-block log-sum-exp routing (Sub-block LSE) preserves more quality than Block mean (Table 3). Block mean reduces indexing time substantially, but attention execution still dominates: construction-time savings are modest at longer context and absent at the shortest context. Sub-block routing is thus a choice to retain quality at an indexing cost. These are comparisons of complete routing rules, including their descriptors, normalization, and aggregation.

Table 3: Routing costs from a separate timing batch. Sum is the reported combined indexing and attention time. Speedup and ΔPSNR are relative to Dense at the same context size.

Context (tokens)	Configuration	Index (ms)	Attn. (ms)	Sum (ms)	Recon. (s)	Speedup	ΔPSNR (dB)
32K	Dense	0	415	415	0.504	1.00×	—
	Sub-block LSE	14	228	242	0.337	1.49×	−0.245
	Block mean	7	228	235	0.342	1.47×	−0.327
64K	Dense	0	1384	1384	1.558	1.00×	—
	Sub-block LSE	43	616	660	0.838	1.86×	−0.203
	Block mean	7	616	623	0.804	1.94×	−0.278
128K	Dense	0	5030	5030	5.385	1.00×	—
	Sub-block LSE	161	1910	2071	2.434	2.21×	−0.152
	Block mean	17	1910	1927	2.289	2.35×	−0.221

Together, these analyses extend the capacity study to inference: the backbone and scene-memory capacity can remain fixed while computation is allocated selectively across interactions and layers. Final-image quality and measured construction time determine the useful allocation. A practical strategy is to preserve early computation and concentrate sparsity in less sensitive later layers, stopping when additional pruning yields little latency reduction relative to its quality cost. This prioritizes useful acceleration over a uniformly high sparsity level.

5 Related Work

Large Reconstruction Model. Feed-forward geometry prediction has progressed from pairwise point maps in DUSt3R (Wang et al., 2024) to joint camera, depth, and point-map estimation in VGGT (Wang et al., 2025). Pi3 (Wang et al., 2026e) removes the privileged reference view, while Depth Anything 3 (Lin et al., 2026) learns a unified depth–ray representation with a plain Transformer. Efficiency is another central direction: FastVGGT (Shen et al., 2026) accelerates inference through token merging, while LiteVGGT (Shu et al., 2026) combines geometry-aware anchor selection with cached merging indices reused across layers. Block-sparse global attention (Wang et al., 2026b) provides a training-free replacement for dense global attention in VGGT, Pi3, and MapAnything. VGGT-Ω (Wang et al., 2026c) improves training scalability through register-mediated cross-view exchange and simplified prediction heads.

Large view synthesis models. LRM (Hong et al., 2024) predicts a neural radiance field from a single image. GS-LRM and Long-LRM (Zhang et al., 2025c; Chen et al., 2025) reconstruct Gaussian scenes from posed images, while Long-LRM++ (Chen et al., 2026) combines feature Gaussians with a learned target-view decoder. AnySplat and YoNoSplat (Jiang et al., 2025b; Ye et al., 2026) support unposed inputs; WorldMirror (Liu et al., 2026) unifies geometry and Gaussian prediction with optional geometric priors. LVSM (Jin et al., 2025) explores latent scene tokens and decoder-only attention, and DVSM (Sun et al., 2026) enables reusable KV memory with shared construction and rendering weights. LaCT (Zhang et al., 2026) represents context through nonlinear fast weights; tttLRM (Wang et al., 2026a) decodes fast-weight scene representations into Gaussians and supports streaming reconstruction. Self-supervised view synthesis offers another route: RayZer (Jiang et al., 2025a) jointly learns cameras and latent scene representations, E-RayZer (Zhao et al., 2026) introduces explicit Gaussian geometry, and RayDer (Prestel et al., 2026) scales static-scene view synthesis training to unconstrained video. SVSM (Kim et al., 2026) studies scaling under matched training compute. Our analysis relates spatial granularity, model capacity, scene memory, and decoding cost; matched training steps do not establish compute-optimal scaling.

Sparse attention. Sparse attention reduces computation by selecting query–key interactions at different granularities. Native Sparse Attention (Yuan et al., 2025) combines compressed, selected, and local attention branches. DeepSeek Sparse Attention (DSA) (DeepSeek-AI, 2025) uses a lightweight learned indexer to select KV tokens, while MoBA (Lu et al., 2025) routes queries to KV blocks through content-based gating. SeerAttention (Gao et al., 2025) learns block-level gates by self-distilling dense attention maps while keeping the pretrained backbone frozen. HiLS-Attention (Hu et al., 2026) learns chunk retrieval through hierarchical attention that combines within-chunk outputs using retrieval scores. Efficient execution provides a complementary direction: SageAttention (Zhang et al., 2025a) accelerates attention through quantization, and SpargeAttention (Zhang et al., 2025b) combines quantized execution with training-free sparse filtering. Token Merging (Bolya et al., 2023) instead reduces computation by aggregating tokens. Building on SpargeAttention's execution backend, we study block routing and layer sensitivity in cross-view reconstruction, focusing on final-image quality and measured reconstruction time while retaining scene-memory capacity.

6 Conclusion and Limitations

Our six findings characterize how network organization, spatial granularity, scene representation, and decoder capacity affect reconstruction quality and cost. They guide the choice of a high-resolution backbone, which serves as a reference for studying sparse cross-view computation. This analysis shows that scene construction can be accelerated while retaining spatial tokens and scene-memory capacity, and relates the achievable gains to layer sensitivity and routing overhead.

Limitations. We do not investigate scaling the amount or diversity of training data, or its interaction with model capacity and scene memory. Our sparse-attention study is limited to inference-time interventions: we do not train or fine-tune the model to adapt to sparse attention patterns, leaving potential gains from such adaptation unexplored.

AI USE STATEMENT

Generative AI assisted with manuscript organization and editing, organization of experimental summaries, numerical consistency checks, and LaTeX preparation.

REPRODUCIBILITY STATEMENT

We have submitted code for the ablation experiments in Table 1 as supplementary material. We will release our training and evaluation code, including experiment configurations and timing scripts. Table 1 identifies the analysis configurations. Section 2 describes the evaluation and resource accounting, and Section 4 defines the backbone and block routing. The appendices detail training protocols, resource estimates, and sparse-inference settings. Evaluation metrics are distinguished from training-curve diagnostics, and comparisons from different checkpoints retain their paired baselines.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3D-aware diffusion models. arXiv preprint arXiv:2304.02602, 2023. URL https://arxiv.org/abs/2304.02602.

Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Fuxin Li, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4349–4359, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Ziwen_Long-LRM_Long-sequence_Large_Reconstruction_Model_for_Wide-coverage_Gaussian_Splats_ICCV_2025_paper.html.

Ziwen Chen, Hao Tan, Peng Wang, Zexiang Xu, and Fuxin Li. Long-LRM++: Preserving fine details in feed-forward wide-coverage reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 370–380, 2026. URL https://openaccess.thecvf.com/content/CVPR2026F/html/Ziwen_Long-LRM_Preserving_Fine_Details_in_Feed-Forward_Wide-Coverage_Reconstruction_CVPRF_2026_paper.html.

DeepSeek-AI. DeepSeek-V3.2-Exp: Boosting long-context efficiency with DeepSeek Sparse Attention. Technical report, 2025. URL https://github.com/deepseek-ai/DeepSeek-V3.2-Exp/blob/main/DeepSeek_V3_2.pdf.

Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T. Barron, Kyle Genova, Nithish Kannen, Sherry Ben, Yandong Li, Mandy Guo, Suhas Yogin, Yiming Gu, Huizhong Chen, Oliver Wang, Saining Xie, Howard Zhou, Kaiming He, Thomas Funkhouser, Jean-Baptiste Alayrac, and Radu Soricut. Image generators are generalist vision learners. arXiv preprint arXiv:2604.20329, 2026. URL https://arxiv.org/abs/2604.20329.

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole. CAT3D: Create anything in 3D with multi-view diffusion models. In *Advances in Neural Information Processing Systems*, 2024. URL https://cat3d.github.io/.

Yizhao Gao, Zhichen Zeng, DaYou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Self-distilled attention gating for efficient long-context prefilling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55846–55869. Curran Associates, Inc., 2025. doi: 10.52202/085713-1869. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50e9dbc4ab68d94f15261ddc26c8ca2b-Abstract-Conference.html.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *International Conference on Learning Representations*, pp. 50678–50702, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/dcad3425f5c8c36b5b3885c091bf1257-Abstract-Conference.html.

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, Kewei Tu, Haitao Mi, and Leo Liang. Hierarchical sparse attention done right: Toward infinite context modeling, 2026. URL https://arxiv.org/abs/2607.02980.

Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, and Georgios Pavlakos. RayZer: A self-supervised large view synthesis model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4918–4929, 2025a. URL https://openaccess.thecvf.com/content/ICCV2025/html/Jiang_RayZer_A_Self-supervised_Large_View_Synthesis_Model_ICCV_2025_paper.html.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: Feed-forward 3D gaussian splatting from unconstrained views. *ACM Transactions on Graphics*, 44(6):1–16, 2025b. doi: 10.1145/3763326. URL https://doi.org/10.1145/3763326.

Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. In *International Conference on Learning Representations*, pp. 60001–60021, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9676c5283df26cabca412ca66b164a7d-Abstract-Conference.html.

Evan Kim, Hyunwoo Ryu, Thomas W. Mitchel, and Vincent Sitzmann. Scaling view synthesis transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28893–28902, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Kim_Scaling_View_Synthesis_Transformers_CVPR_2026_paper.html.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations*, pp. 141261–141285, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e4cd50120b6d7e8daff1749d6bbaa889-Paper-Conference.pdf.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. 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*, pp. 22160–22169, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Ling_DL3DV-10K_A_Large-Scale_Scene_Dataset_for_Deep_Learning-based_3D_Vision_CVPR_2024_paper.html.

Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. WorldMirror: Universal 3D world reconstruction with any-prior prompting. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=HFNJOpXHfm.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17790–17815. Curran Associates, Inc., 2025. doi: 10.52202/085713-0602.

URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19eae75beed66321d62272e794a9c2ac-Abstract-Conference.html.

Ziqiao Ma, Xuweiyi Chen, Shoubin Yu, Sai Bi, Kai Zhang, Ziwen Chen, Sihan Xu, Jianing Yang, Zexiang Xu, Kalyan Sunkavalli, Mohit Bansal, Joyce Chai, and Hao Tan. 4D-LRM: Large space-time reconstruction model from and to any view at any time. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2730. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/753d9584b57ba01a10482f1ea7734a89-Abstract-Conference.html.

Ziqiao Ma, Xueyang Yu, Haoyu Zhen, Yuncong Yang, Joyce Chai, and Chuang Gan. Fast spatial memory with scalable elastic test-time training. In *Computer Vision – ECCV 2026*, volume 17078 of *Lecture Notes in Computer Science*, pp. 217–236. Springer, 2026. doi: 10.1007/978-3-032-37393-9_14. URL https://link.springer.com/chapter/10.1007/978-3-032-37393-9_14.

Ulrich Prestel, Stefan Andreas Baumann, Nick Stracke, and Björn Ommer. RayDer: Scalable self-supervised novel view synthesis from real-world video. In *Computer Vision – ECCV 2026*, volume 17012 of *Lecture Notes in Computer Science*, pp. 558–578. Springer, 2026. doi: 10.1007/978-3-032-37044-0_31. URL https://link.springer.com/chapter/10.1007/978-3-032-37044-0_31.

You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. FastVGGT: Fast visual geometry transformer. In *International Conference on Learning Representations*, pp. 98040–98062, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9f5b159f2c1d216d73bc0c289763dbfc-Paper-Conference.pdf.

Zhijian Shu, Cheng Lin, Tao Xie, Wei Yin, Ben Li, Zhiyuan Pu, Weize Li, Yao Yao, Xun Cao, Xiaoyang Guo, and Xiao-Xiao Long. LiteVGGT: Boosting vanilla VGGT via geometry-aware cached token merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36422–36432, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Shu_LiteVGGT_Boosting_Vanilla_VGGT_via_Geometry-aware_Cached_Token_Merging_CVPR_2026_paper.html.

Cheng Sun, Jaesung Choe, Min-Hung Chen, Ryo Hachiuma, and Yu-Chiang Frank Wang. DVSM: Decoder-only view synthesis model done right, 2026. URL https://arxiv.org/abs/2605.29891.

Chen Wang, Hao Tan, Wang Yifan, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttLRM: Test-time training for long context and autoregressive 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36582–36592, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_tttLRM_Test-Time_Training_for_Long_Context_and_Autoregressive_3D_Reconstruction_CVPR_2026_paper.html.

Chung-Shien Brian Wang, Christian Schmidt, Jens Piekenbrinck, and Bastian Leibe. Block-sparse global attention for efficient multi-view geometry transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14546–14555, 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_Block-Sparse_Global_Attention_for_Efficient_Multi-View_Geometry_Transformers_CVPR_2026_paper.html.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5294–5306, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VGGT_Visual_Geometry_Grounded_Transformer_CVPR_2025_paper.html.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21486–21499, 2026c. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_VGGT-ohm_CVPR_2026_paper.html.

Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, Andrei Zanfir, and Cristian Sminchisescu. Video generation models are general-purpose vision learners. In *Computer Vision – ECCV 2026*, volume 17005 of *Lecture Notes in Computer Science*, pp. 608–627. Springer, 2026d. doi: 10.1007/978-3-032-37369-4_33. URL https://link.springer.com/chapter/10.1007/978-3-032-37369-4_33.

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*, pp. 20697–20709, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Wang_DUSt3R_Geometric_3D_Vision_Made_Easy_CVPR_2024_paper.html.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations*, pp. 10481–10497, 2026e. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/11a09e0aaa74867c6b0719c639fc09f8-Paper-Conference.pdf.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. YoNoSplat: You only need one model for feedforward 3D gaussian splatting. In *International Conference on Learning Representations*, pp. 39852–39871, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/42f92b78a6695f60db0cd38b54d57a41-Abstract-Conference.html.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Jintao Zhang, Jia Wei, Pengle Zhang, Jun Zhu, and Jianfei Chen. SageAttention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, pp. 71566–71585, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/b286c344d38e10d2466c0514b78e2f36-Abstract-Conference.html.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 76397–76413, 2025b. URL https://proceedings.mlr.press/v267/zhang25ch.html.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3D gaussian splatting. In *Computer Vision – ECCV 2024*, volume 15080 of *Lecture Notes in Computer Science*, pp. 1–19. Springer, 2025c. doi: 10.1007/978-3-031-72670-5_1. URL https://doi.org/10.1007/978-3-031-72670-5_1.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *International Conference on Learning Representations*, pp. 157604–157638, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ffbfff81fb78e4bb558273b91e12e318-Abstract-Conference.html.

Qitao Zhao, Hao Tan, Qianqian Wang, Sai Bi, Kai Zhang, Kalyan Sunkavalli, Shubham Tulsiani, and Hanwen Jiang. E-RayZer: Self-supervised 3D reconstruction as spatial visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7525–7535, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhao_E-RayZer_Self-supervised_3D_Reconstruction_as_Spatial_Visual_Pre-training_CVPR_2026_paper.html.

A SCENE REPRESENTATIONS AND CAPACITY

We provide the training setup, forward paths, resource definitions, and training trajectories for the representation study in Table 1 and Section 3. We have submitted code for the ablation experiments in Table 1 as supplementary material.

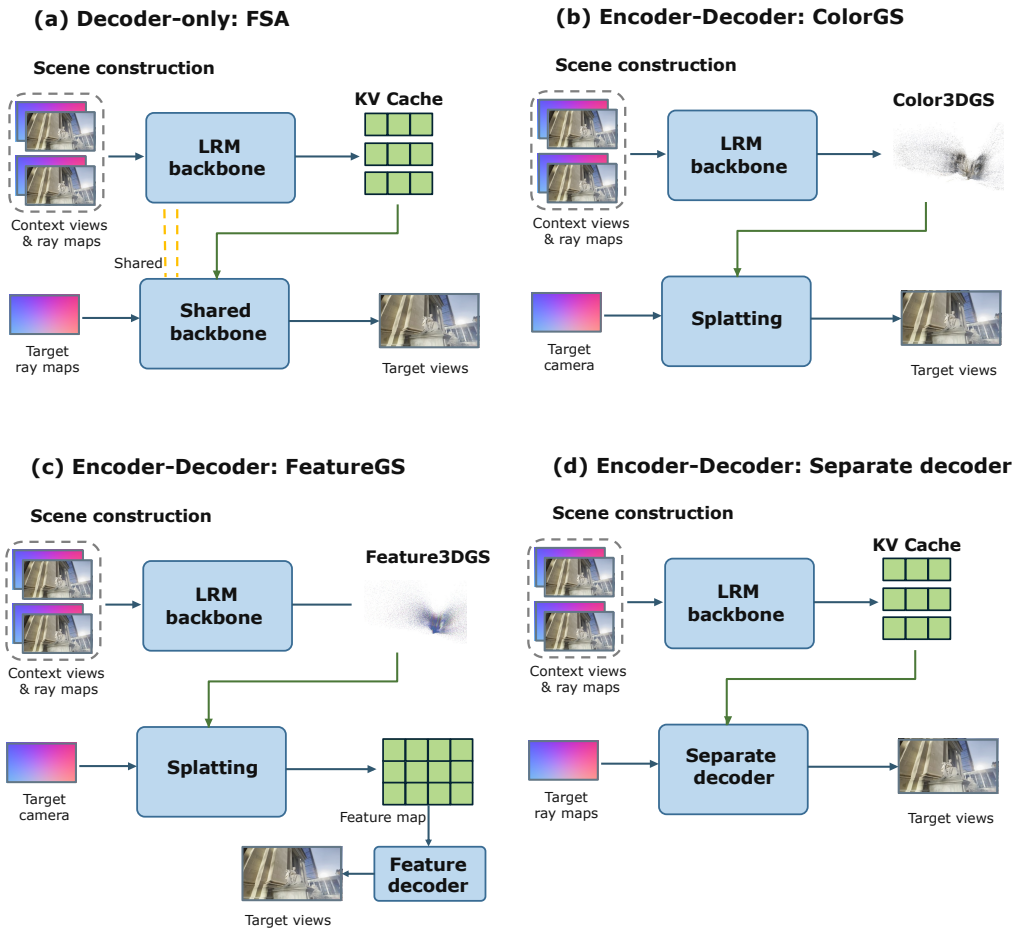

Figure 7: Scene construction and target-view synthesis in four model families. (a) Decoder-only FSA shares a backbone between context and target passes, retaining layerwise KV memory. (b) Color GS splats directly to RGB. (c) FeatureGS splats features before learned decoding. (d) A separate decoder queries KV memory using target rays. The dashed yellow connection denotes shared weights.

A.1 TRAINING SETUP

The architectural study uses 144×256 images, 32 context views, and 32 target views, which may include context views. Training uses global batch 32, learning rate 4×10^{-4}, 4K warmup steps, and a 50K-step budget. Baseline runs use eight GPUs; the KV sweep uses four at the same global batch. The objective is MSE plus $0.2\times$LPIPS, with $0.01\times$opacity regularization for Gaussian variants. Training curves include only logged steps; the six representative systems in Figure 1 and all KV capacity runs have complete 50K-step logs. Training-loss curves are optimization diagnostics, distinct from test-image quality measurements.

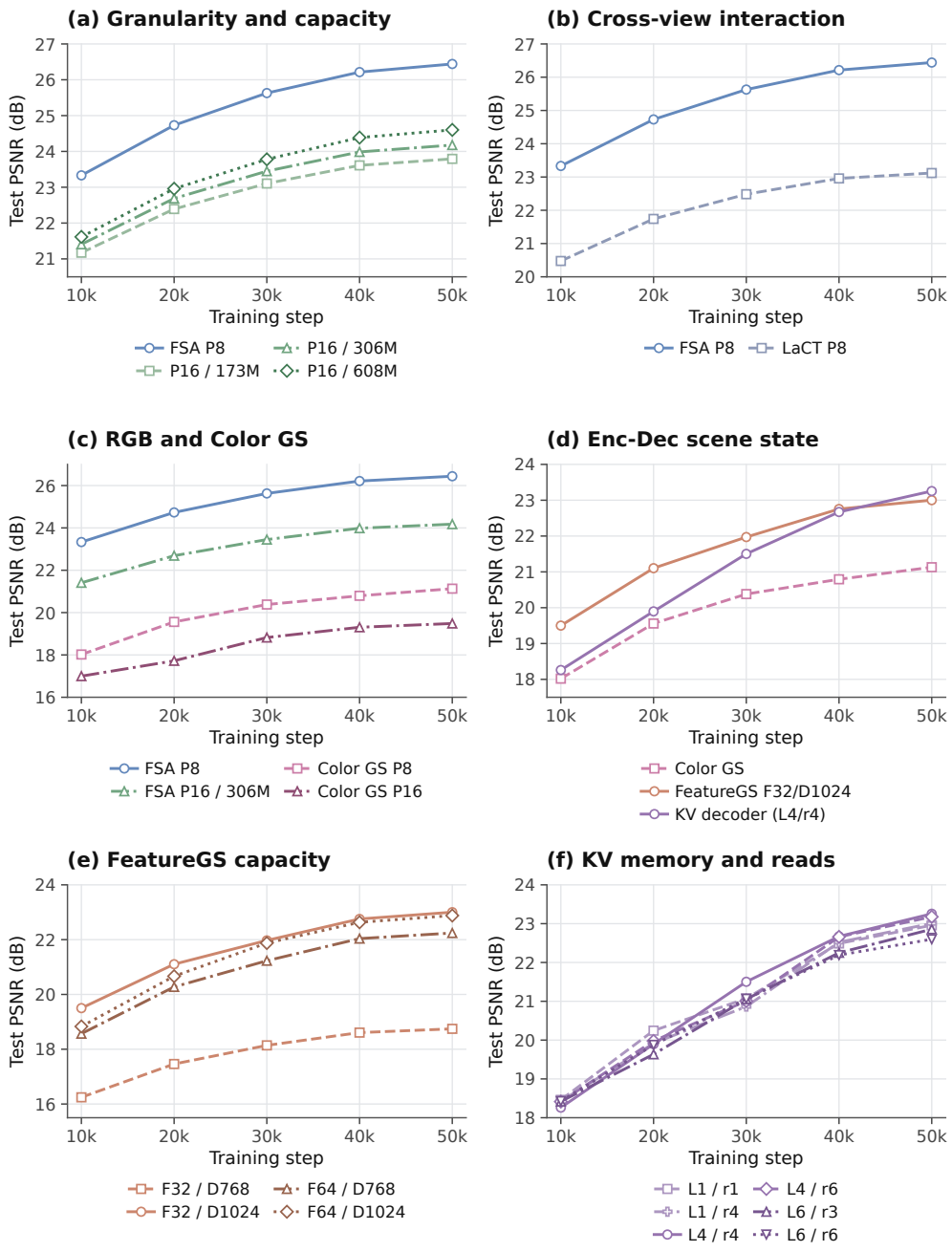

Figure 8: Test PSNR throughout training for 17 configurations, evaluated every 10K steps on 131 scenes and 5,822 target views. Markers denote measured checkpoints. Panel (c) includes the additional P16/D1024 Color GS model. In (e), F/D denote feature/decoder width; F32/D1024 has six MLPs versus eight in the other FeatureGS runs. In (f), L/r denote retained KV stages/query depth. Panels use different PSNR ranges to resolve within-group differences.

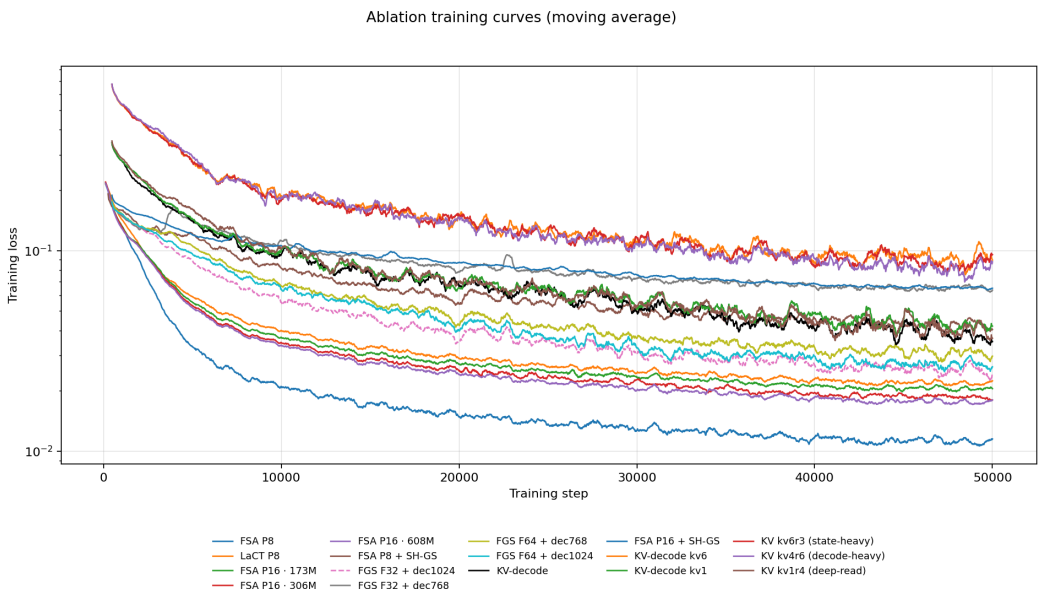

Figure 9: Training-loss curves across attention, Gaussian, and KV-decoder configurations. Curves are moving averages, with a logarithmic loss axis. Logging and regularization differ across families; these curves are optimization diagnostics, not a cross-family quality ranking.

A.2 Forward Paths and Scene Representations

Figure 7 summarizes the four construction and query paths. A pixel with ray origin $\mathbf{o}$ and unit direction $\mathbf{d}$ uses the camera encoding $\rho = [\mathbf{o}, \mathbf{d}, \mathbf{o} \times \mathbf{d}]$. Context patches contain $[\rho, 2I - 1]$; target queries in the shared backbone use zero RGB channels.

Decoder-only RGB prediction. FSA stores context keys and values at each cross-view stage. Target-ray tokens query the corresponding state through the same patch embedding and backbone, then map to RGB patches. LaCT uses the same shared construction and query organization but replaces KV attention with a multi-head fast-weight function. Its scene-specific weights are updated from context tokens and held fixed during target querying.

Color GS. Pose-only queries at the *context* cameras are appended to the posed RGB tokens. A Gaussian head maps their output features to geometry, opacity, and spherical-harmonic color coefficients. The resulting Gaussians form the scene state; target cameras condition splatting directly to RGB.

FeatureGS. The context backbone predicts Gaussians carrying F-dimensional features alongside geometry and opacity. Target-camera rasterization produces a feature map Z_t. A learned decoder embeds patches of $[Z_t, \rho_t]$ with two-dimensional positional encodings and predicts RGB through local and full-image spatial processing, independently within each target view. Thus Color GS and FeatureGS differ in both scene construction and target decoding, as well as in their stored attributes.

KV memory with a separate decoder. The context backbone retains KV tensors from L cross-view stages. These are concatenated along the head/channel dimension while preserving the context-token axis. Target-ray patches initialize a separate decoder whose r cross-attention layers read the *same concatenated state*, each with its own projections. Local and full-image refinement precede the RGB head. Here L controls memory and reader input width, whereas r controls query depth; the two are independent configuration choices.

A.3 Resource Accounting

Network parameters. Shared network parameters are counted once. In Table 1, context backbones have width 768 and 12 blocks, except the 306M and 608M P16 models, which use width 1024 with 12 and 24 blocks, respectively. Figure 8(c) additionally includes a P16 Color GS model with width 1024.

Persistent scene memory. For persistent tensors T_i with b_i bytes per scalar, scene memory is

$$M_{\text{state}} = \sum_i b_i |T_i|, \tag{5}$$

where $|T_i|$ is the number of scalars. Shared model weights and temporary buffers are excluded. KV and Gaussian payloads use a common BF16 convention ($b_i = 2$), independently of runtime dtype. For N_c context tokens, hidden width D, and L retained KV stages, the payload is $4LN_cD$ bytes.

For one Gaussian per context pixel, the BF16 payload is $2V_cHWd$ bytes, where $d = 59$ for Color GS (SH degree 3) and $d = 11 + F$ for FeatureGS. LaCT memory counts its scene-specific fast weights.

Computation and timing. FLOPs are reported per scene sample at 144×256 with 32 context and 32 target views, counting two operations per multiply–add. Training FLOPs are estimated as three times the forward count. These are model-compute estimates, not measured end-to-end training costs; Gaussian splatting is excluded. Construction and target rendering are timed separately.

A.4 Capacity Sweeps and Evaluation

KV configurations. The six runs use a P8, width-768, 12-block backbone, seed 95, and the training setup above. Their (L, r) pairs are $(1, 1)$, $(1, 4)$, $(4, 4)$, $(4, 6)$, $(6, 3)$, and $(6, 6)$. Each retains the last L KV stages, giving reader input width $768L$ and BF16 payload $54L$ MiB. The sweep provides comparisons at fixed memory or query depth, rather than a full factorial design.

FeatureGS configurations. The four FeatureGS runs combine $F \in \{32, 64\}$ with decoder widths $D_{\text{dec}} \in \{768, 1024\}$. Each decoder has one local and one full-image attention module, with residual MLPs in both stages. F32/D1024 uses 3 local and 5 global modules, totaling 2 attention and 6 MLP modules; the other three use 4 local and 6 global modules, totaling 2 attention and 8 MLP modules. Thus comparisons involving F32/D1024 also change refinement depth. The KV sweep separately keeps 4 local and 4 global refinement modules fixed.

Evaluation. The KV capacity study follows the evaluation protocol of Long-LRM (Chen et al., 2025), using 144×256 images, 32 context views, and fold-8 KMeans selection. Quality is averaged per target over 131 scenes after excluding two warmup scenes. KV timing uses one scene per batch on an A100 with BF16 autocast.

Figure 8 evaluates 17 configurations every 10K steps on 131 scenes and 5,822 target views. The 50K-step PSNR values match the corresponding entries in Table 1. Table 1's rendering FPS comes from separate timing measurements.

B HIGH-RESOLUTION BACKBONE AND SPARSE ATTENTION

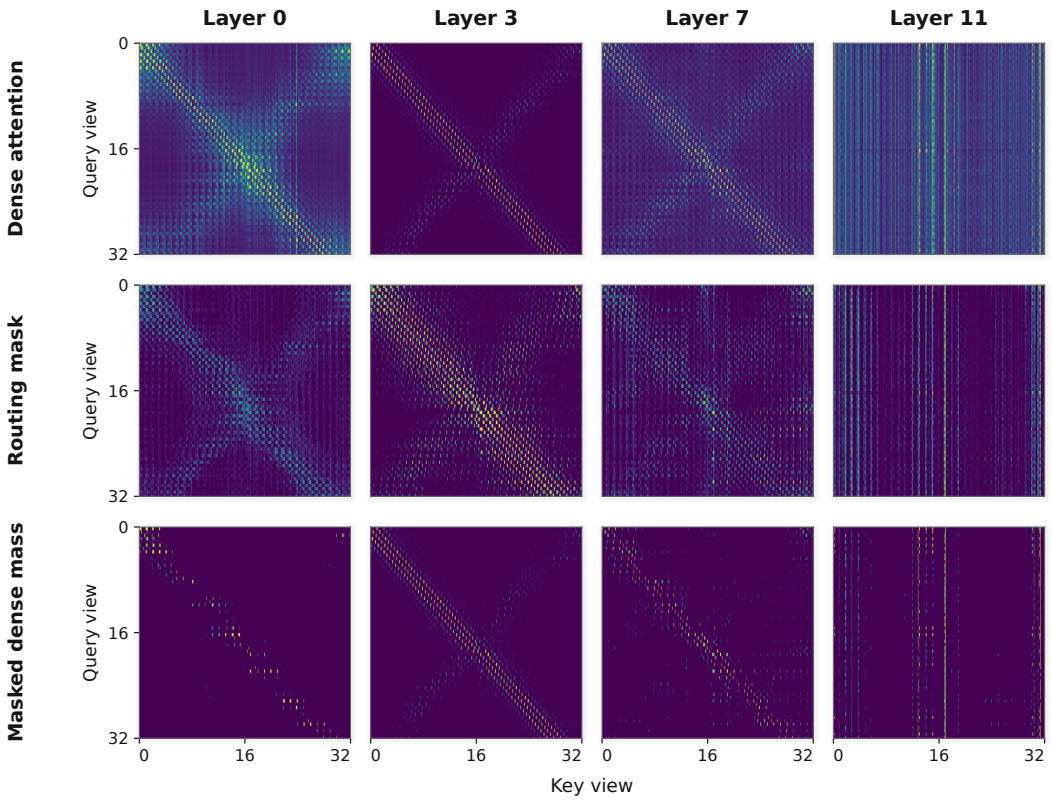

Figure 10: Cross-view attention at four depths for one scene with 32 context views (65,280 tokens). Columns use zero-based layer indices; rows show dense attention mass, the routing mask at 6.25% retention, and masked dense attention mass. Dense mass is averaged over heads. Each panel displays 64 of 510 query blocks, sampled uniformly, and all 1,020 key blocks in view order. Colors use independent 99.5th-percentile normalization, so brightness is not directly comparable across panels.

Figure 10 shows attention at four depths for scene `032dee9fb0a8` using the final checkpoint from stage 4. Layer 3 has concentrated bands; layers 0 and 7 are more diffuse, and layer 11 exhibits vertical stripes. These spatial patterns complement the reconstruction-sensitivity measurements in Figure 4(c).

B.1 BACKBONE AND TRAINING

The 306M-parameter BVSM backbone uses patch size 16, width 1024, and 16 heads. Its 12 blocks pair within-view and cross-view attention, giving 24 attention operations. Queries and keys receive per-head RMS normalization. At 544×960, each view contains 2,040 tokens, so 16/32/64 context views correspond to approximately 32K/64K/128K tokens.

Training follows four successive stages (Table 4), increasing resolution before randomly sampling 16, 32, or 48 context views in the final stage. Stage 4 ends at 8K steps, giving 148K steps in total; we evaluate this final checkpoint. Sparse inference modifies context cross-view attention; target rendering remains dense and reads the full layerwise KV cache.

Table 4: Progressive training of the BVSM backbone. Resolution is $H \times W$; steps are reported separately for each stage.

Stage	Resolution	Global batch	Learning rate	Context views	Target views	Steps
1	128×256	512	4×10^{-4}	32	32	100K
2	256×480	512	1×10^{-4}	32	32	30K
3	544×960	64	5×10^{-5}	32	32	10K
4	544×960	64	1×10^{-5}	$\{16, 32, 48\}$	32	8K

B.2 Sparse Attention Implementation

Blocks and budgets. Query and key blocks contain $b_Q = 128$ and $b_K = 64$ tokens. They follow the flattened context sequence and may cross view boundaries. With $B_K = \lceil N_K / b_K \rceil$ key blocks and ties-to-even rounding, each query block selects

$$k_\ell = \min\{B_K, \max\{1, \mathrm{round}(\rho_\ell B_K)\}\}. \tag{6}$$

The *BVSM-Sparse* configuration in Table 2 uses

$$\rho_\ell = \begin{cases} 1, & \ell \in \{0, 1\}, \\ 1/2, & \ell \in \{2, 3\}, \\ 1/16, & \ell \in \{4, \ldots, 11\}. \end{cases} \tag{7}$$

Layer indices count the 12 cross-view modules. Retention ρ_ℓ denotes the selected block-interaction fraction, rather than a KV-storage or end-to-end FLOPs ratio.

Routing. Mean pooling over contiguous 32-token sub-blocks yields four query and two key descriptors per block. Descriptors are converted to FP32 and ℓ_2-normalized before the log-sum-exp scoring in Equation 3. Incomplete blocks are zero-padded for pooling; execution retains actual sequence lengths. Top-k_ℓ selection is independent for each query block, head, and layer. The Block mean baseline instead uses one unnormalized mean per block and scaled dot-product scoring.

Execution. Indexing processes at most $C = 256$ query blocks per chunk, reducing temporary score storage from $\mathcal{O}(8B_Q B_K)$ to $\mathcal{O}(8CB_K)$ entries per head, where $B_Q = \lceil N_Q / b_Q \rceil$. The selected-block mask is converted to a key-block lookup table and valid-block counts. SpargeAttention's SM80 backend (Zhang et al., 2025b) on A100 uses INT8 queries/keys with blockwise scales and FP16 values, without materializing dense token-level attention. Retention one uses the original dense path. Reported sparse results include both block selection and quantized execution.

Local token merging. Our token-merging alternatives aggregate context tokens within each view using spatial groups or local anchors based on feature similarity and position. In the attention-internal variant, attention and its output projection operate on compact tokens, then outputs are mapped back to the original layout before subsequent MLPs and blocks. Hard assignments broadcast compact outputs to their groups; soft assignments use weighted combinations. Restoring the sequence dimensions does not recover the original features. Grouping is query-independent, whereas block-sparse routing selects interactions independently for each query block.

B.3 Evaluation Protocols

Main evaluation. We follow the evaluation protocol of Long-LRM (Chen et al., 2025) on DL3DV: fold-8 KMeans selection, 133 valid scenes, 5,901 targets, and 16/32/64 context views. Dense and Sparse share the checkpoint and inputs. Reconstruction and target rendering are timed separately for a single scene on an A100; Table 6 reports the paired measurements.

Retention sweep and baselines. Figure 5(b) varies retention in layers 4–11 with layers 0–3 dense, at 32 context views. This differs from Equation 7, which also sparsifies layers 2–3. In panel (d), block sparsity and token merging use layers 6–11 at 25% retention; attention merging uses layers 4–11 at 25%, and cache pruning retains 75%. All use the same checkpoint. These budgets refer to different resources and do not imply matched computation.

Layer sensitivity and profiling. Figure 4(c) uses six scenes, 16 targets per scene, and 32 context views, with a separate Dense baseline of 29.143 dB. Only the indicated cross-view layer retains 6.25%; all others remain dense. Figure 5(a) profiles modules from an earlier checkpoint independently of the paired runtime evaluation; its displayed shares are not renormalized.

C Additional Experimental Results

We complement the high-resolution evaluation with model size, rendering throughput, peak memory, and qualitative comparisons.

C.1 Parameters, Rendering Throughput, and Memory

Cross-method comparison. Table 5 adds model size and rendering throughput to Table 2. Reference runtimes retain their source measurement conditions. LVSM and LaCT parameter counts follow Sun et al. (2026); LongLRM and LongLRM++ use their respective papers (Chen et al., 2025; 2026). The approximately 400M tttLRM count is derived from its official full-model configuration (Wang et al., 2026a), including fast-weight initializations and the Gaussian head but excluding training-loss networks.

Table 5: Model parameters and rendering throughput for the methods in Table 2. FPS values retain their source measurement conditions and are shown as integers; > 50 entries are highlighted together. Parameters exclude scene-specific state; 3DGS is scene-dependent. Dense/Sparse share parameters; FPS is derived from Table 6.

Method	Params.	Render FPS↑		
		16 input views	32 input views	64 input views
3DGS	—	$\mathbf{> 50}$	$\mathbf{> 50}$	$\mathbf{> 50}$
LVSM	165M	20	20	19
Long-LRM	142M	$\mathbf{> 50}$	$\mathbf{> 50}$	$\mathbf{> 50}$
Long-LRM++	110M	14	14	14
tttLRM	400M	$\mathbf{> 50}$	$\mathbf{> 50}$	$\mathbf{> 50}$
LaCT	379M	2	2	2
BVSM-Dense	306M	25	15	9
BVSM-Sparse	306M	25	15	9

Paired Dense and Sparse measurements. Table 6 reports reconstruction time, target rendering, and peak memory at fixed inputs. It uses the main evaluation protocol in Appendix B.3; the independently timed router comparison remains in Table 3.

Table 6: Paired BVSM-Dense and BVSM-Sparse runtime measurements for Table 2. Reconstruction and rendering times retain the reported precision. Speedup is relative to Dense at the same input-view count. Peak memory is unchanged at the reported precision. FPS is computed as $1000/t_{\text{render}}$ from the reported time in milliseconds per view and rounded to the nearest integer. Peak GB is the reported runtime memory, not the logical scene-memory payload.

Input views	Method	Recon. (s)	Speedup	Render (ms/view)	FPS	Peak (GB)
16	BVSM-Dense	0.505	—	40.4	25	5.5
16	BVSM-Sparse	0.334	1.51×	40.0	25	5.5
32	BVSM-Dense	1.562	—	65.9	15	8.9
32	BVSM-Sparse	0.838	1.86×	65.6	15	8.9
64	BVSM-Dense	5.410	—	116.6	9	15.6
64	BVSM-Sparse	2.432	2.22×	116.0	9	15.6

C.2 Qualitative Effects of Sparsity

Figure 11 extends Figure 6 with two target views, showing how retention and the choice of sparse layers affect reconstruction and error relative to ground truth.

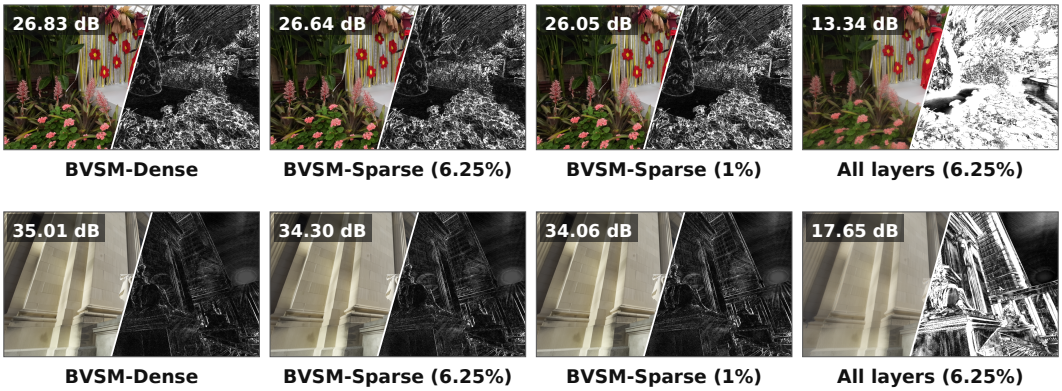

Figure 11: Additional qualitative comparisons at two target views. Each row compares BVSM-Dense, BVSM-Sparse at 6.25% and 1% retention, and 6.25% retention in all context cross-view layers. Each panel shows the reconstruction on the left and error relative to ground truth on the right. Inset PSNR values correspond to the displayed target view.

D Further Conclusion

Temporal scene memory. Extending scene memory over time is a promising direction beyond our static-scene study. 4D-LRM learns a unified space-time representation for synthesis at novel views and times, while Fast Spatial Memory incorporates elastic test-time updates to retain information across dynamic observations (Ma et al., 2025; 2026). These developments motivate studying how spatial capacity and temporal updates should be allocated jointly. Although we do not investigate this dimension here, understanding what to retain, update, and query in an evolving scene memory could help scale LVSMs to longer dynamic sequences.

NVS as representation learning. RayZer and E-RayZer connect self-supervised view synthesis with spatial representation learning (Jiang et al., 2025a; Zhao et al., 2026). More broadly, recent image and video generation models transfer generative pretraining to visual perception, including geometry estimation (Gabeur et al., 2026; Wang et al., 2026d). This raises a further question: how far can NVS serve as a scalable pretraining objective for general visual representations? An important next step is to examine whether design choices that improve reconstruction also produce scene features that transfer to downstream geometry and semantic understanding.

Generative reconstruction and scene memory. NVS is closely connected to generative reconstruction: sparse observations constrain visible content but leave unseen regions ambiguous. GeNVS and CAT3D use diffusion models to generate novel views conditioned on observed images (Chan et al., 2023; Gao et al., 2024); CAT3D further uses the generated views for 3D reconstruction. These works illustrate how generation can complement reconstruction by synthesizing plausible content beyond the observations. From the scene-memory perspective studied here, we ask whether memory learned through NVS could also provide persistent, scene-specific conditioning for video generation. Such memory could retain information about previously observed regions, while a generative model synthesizes unseen content. Whether this combination can maintain 3D consistency across viewpoints and over time is a promising direction that we leave for future work.

E Qualitative Comparison with Other Methods

Figure 12 compares BVSM with tttLRM and LaCT on eight scenes at 32 context views, complementing the quantitative results in Table 2.

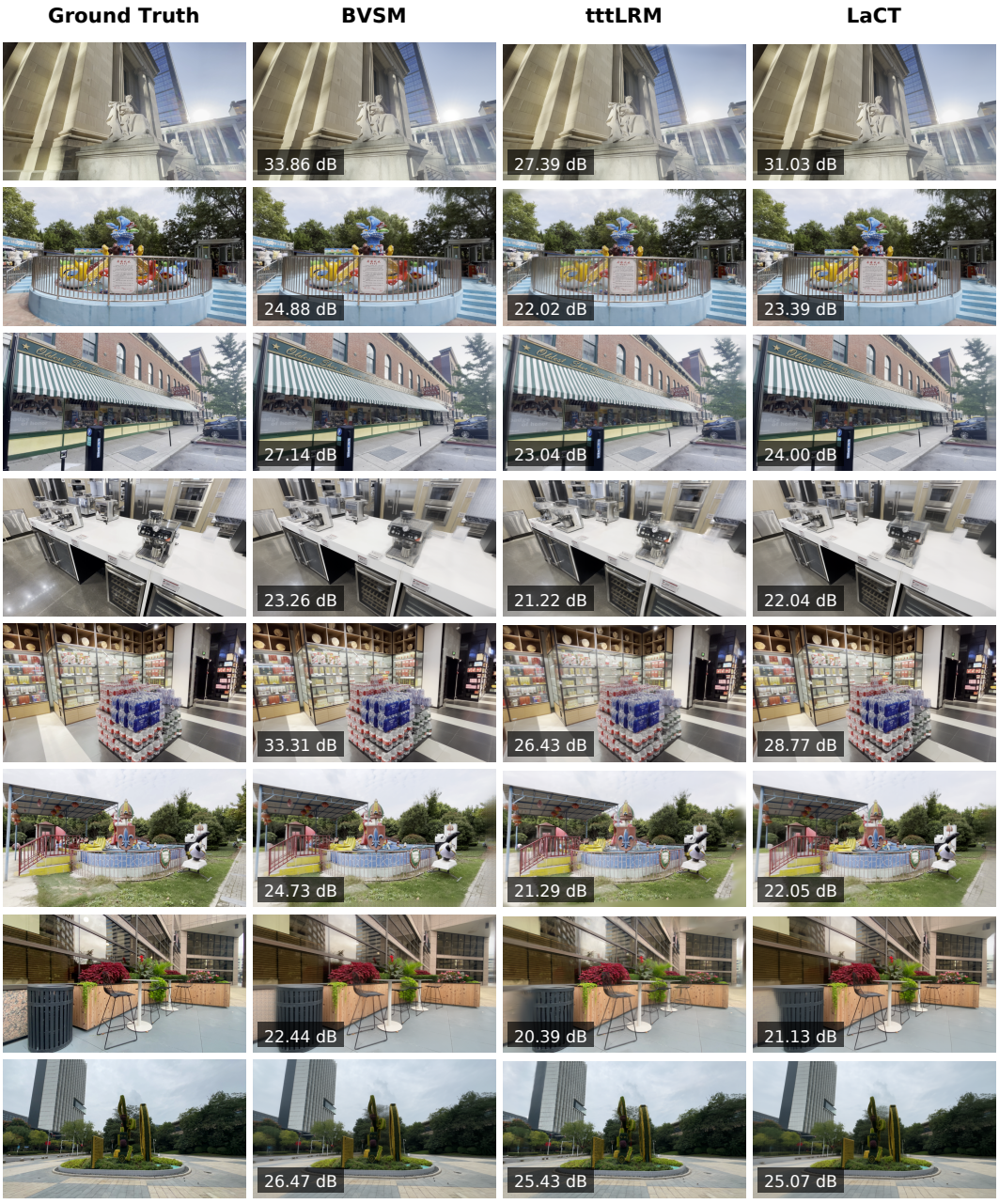

Figure 12: Qualitative comparison of BVSM, tttLRM, and LaCT with 32 context views across eight scenes, using the first Long-LRM target frame in each scene. Insets report per-image PSNR at each method's native resolution: 544×960 for BVSM and 536×960 for tttLRM and LaCT. The ground-truth column shows the 544×960 reference; images retain their native aspect ratios.