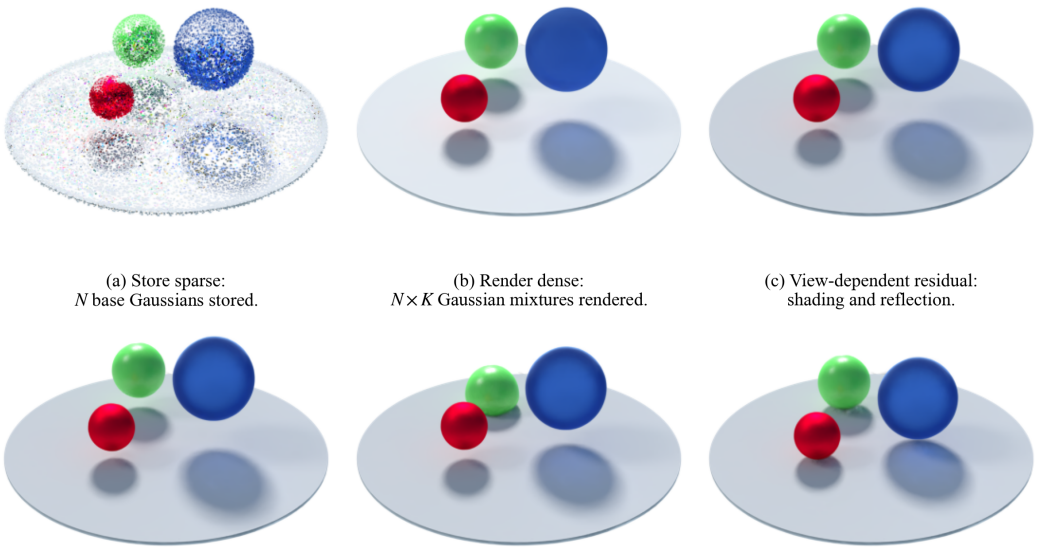
STORE SPARSE, RENDER DENSE: NEURAL GAUSSIAN MIXTURE SPLATTING

Anonymous authors
Paper under double-blind review



(a) Store sparse: N base Gaussians stored.

(b) Render dense: $N \times K$ Gaussian mixtures rendered.

(c) View-dependent residual: shading and reflection.

(d) Time-dependent residual: motion and deformation.

Figure 1: **Store sparse, render dense.** Neural Gaussian Mixture Splatting (NGMS) stores N complete base Gaussians **(a)** and expands each into K residual components at render time, producing NK ordinary Gaussians for an unmodified rasterizer **(b)**. A factorized residual field predicts view-dependent shading and reflection **(c)** and, for dynamic scenes, motion and deformation **(d)**.

ABSTRACT

We introduce Neural Gaussian Mixture Splatting (NGMS), a compact, unified representation for static and dynamic scenes. Our core insight is that dense geometry, view-dependent appearance, and motion can be generated from a sparse set of stored Gaussians through a shared residual field. NGMS stores N complete base Gaussians and expands each into K local components through learned residuals in position, rotation, scale, opacity, and spherical-harmonic (SH) appearance. A single nonlinear field query per base jointly predicts all K components, reducing nonlinear evaluations by a factor of K relative to independent per-component prediction. Only diffuse base Gaussians and shared network weights are stored, avoiding independent storage of every rendered primitive and its higher-order SH coefficients. Factorized view and time branches capture appearance and deformation within the same formulation, while the generated components retain compatibility with standard Gaussian rasterization. Across four static and dynamic benchmarks, NGMS achieves the smallest serialized models among the evaluated methods, with competitive static reconstruction quality and a 6.5 dB PSNR gain over 4DGS on D-NeRF. Dataset-average storage is reduced by 9-28$\times$ relative to scaffold methods and by 31-303$\times$ relative to explicit 3D/4DGS, reaching two orders of magnitude on dynamic scenes while maintaining real-time rendering. Code is available at our project page: `https://www.example.com`.

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

1 INTRODUCTION

Compact, high-quality scene representations are essential for practical novel-view synthesis. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) achieves real-time photorealistic rendering through explicit primitives and efficient differentiable rasterization. Its representation presents three challenges. First, fine geometry and repeated structures are encoded by dense, independently stored splats. Second, view-dependent appearance requires spherical-harmonic (SH) coefficients for every primitive, increasing storage (Lee et al., 2024a; Fan et al., 2024). Third, vanilla 3DGS is static, with no mechanism for motion or deformation. Addressing these challenges together would make Gaussian rendering both more compact and more expressive.

Shared neural functions represent scene detail compactly, as demonstrated by static and dynamic neural fields (Mildenhall et al., 2020; Pumarola et al., 2021). Applying a residual network independently to every rendered Gaussian, however, makes nonlinear computation grow with the full primitive count. This cost is particularly relevant to dynamic Gaussian methods that use neural deformation fields to update large primitive sets (Yang et al., 2024c; Wu et al., 2024).

Scaffold-GS addresses this cost through sparse feature-bearing anchors, jointly decoding K view-adaptive candidate Gaussians from each visible anchor (Lu et al., 2024a). Each anchor stores a latent feature vector and K explicit offsets to support local generation. We instead retain complete Gaussians as the stored structure and learn local detail through a shared function.

Our core insight is that shared residual generation can address geometry, appearance, and dynamics within one Gaussian representation. We introduce *Neural Gaussian Mixture Splatting* (NGMS), which stores N renderable base Gaussians and uses a shared residual field to generate K local components around each base. The nonlinear spatial and conditioning networks are evaluated once per base, and lightweight linear heads jointly predict all K components. This reduces nonlinear field queries by a factor of K relative to independent per-component prediction, while preserving NK Gaussians for rendering. Only base Gaussian attributes and shared network weights are stored.

This formulation addresses the three challenges directly. Shared local mixtures generate geometric detail from the stored bases; shared SH prediction adds view-dependent appearance to diffuse base colors; and an optional time-conditioned branch adds motion and deformation. A view-conditioned branch adapts both geometry and appearance to the camera. Component opacities are learned as residuals around the base opacity and normalized by the mixture size. The generated components remain ordinary Gaussians, with their view-conditioned SH coefficients evaluated by the standard rasterizer. This compatible design further reduces storage by combining Gaussian quantization and spatial ordering (Niantic Labs, 2024; Tian et al., 2025; Morgenstern et al., 2024) with reduced-precision shared-field weights.

We validate NGMS on NeRF Synthetic (Mildenhall et al., 2020), Mip-NeRF 360 (Barron et al., 2022), D-NeRF (Pumarola et al., 2021), and Neural 3D Video (Li et al., 2022), evaluating reconstruction quality, serialized model size, and rendering speed. NGMS achieves competitive static reconstruction quality and improves PSNR by $6.5\,$dB over 4DGS on D-NeRF, while maintaining real-time rendering. It produces the smallest models among the evaluated methods on all four benchmarks, reducing dataset-average storage by 9-28$\times$ relative to scaffold representations and by 31-303$\times$ relative to explicit 3D/4DGS.

Our contributions are:

- **Gaussian-native residual mixtures.** A compact representation that combines complete base Gaussians with a shared field to generate local geometry and appearance, replacing per-base latent vectors and offset tables.
- **Sparse neural queries, dense rasterization.** Joint prediction of K components from each base Gaussian reduces nonlinear evaluations by a factor of K relative to independent per-component models while preserving dense Gaussian rendering.
- **Compact and unified representation.** Factorized view and time branches capture appearance and motion within a single residual formulation, achieving the smallest evaluated models on four benchmarks with competitive static quality and a $6.5\,$dB gain over 4DGS on D-NeRF.

2 Related Work

Neural radiance fields. Neural Radiance Fields (NeRF) (Mildenhall et al., 2020) represent scenes as shared continuous functions for novel-view synthesis. Extensions improve quality and efficiency through anti-aliasing, hash encodings, low-rank factorization, voxel grids, octrees, and baking (Barron et al., 2021; 2023; Müller et al., 2022; Chen et al., 2022; Sun et al., 2022; Yu et al., 2021; Hedman et al., 2021). Dynamic fields use time conditioning, deformation, or space-time factorization (Pumarola et al., 2021; Park et al., 2021a;b; Fridovich-Keil et al., 2023; Cao & Johnson, 2023; Li et al., 2022; Lou et al., 2024). NGMS combines neural sharing with explicit Gaussian generation and rasterization.

Explicit Gaussian splatting. Building on classical point and volume splatting (Zwicker et al., 2001b;a), 3DGS (Kerbl et al., 2023) enables real-time rendering with optimized anisotropic Gaussians and differentiable rasterization. Extensions improve anti-aliasing, density control, optimization, and rasterization efficiency (Yu et al., 2024; Yan et al., 2024; Zhang et al., 2024; Rota Bulò et al., 2024; Kheradmand et al., 2024; Feng et al., 2025). Other work enriches geometry and appearance through alternative primitives (Huang et al., 2024a; Hamdi et al., 2024; Held et al., 2025; 2026), adaptive Beta kernels (Liu et al., 2025; 2026b), and textured or directional appearance models (Chao et al., 2025; Yang et al., 2024b; Gao et al., 2025a; Diolatzis et al., 2024). NGMS instead generates local detail through a shared residual field to reduce per-primitive storage.

Dynamic Gaussian splatting. Dynamic Gaussian methods track persistent primitives, learn deformation fields, or optimize spacetime attributes (Luiten et al., 2024; Yang et al., 2024c;a; Wu et al., 2024; Lin et al., 2024; Lu et al., 2024b; Lee et al., 2024b; Li et al., 2024). Further work introduces sparse motion control, motion-aware optimization, and joint spatial-temporal-angular primitives (Guo et al., 2025; Huang et al., 2024b; Gao et al., 2025b). NGMS unifies static and dynamic scenes with optional time-conditioned geometry and appearance residuals around persistent Gaussian bases.

Scaffold-based Gaussian representations. Scaffold-GS (Lu et al., 2024a) generates view-adaptive Gaussians from anchors storing latent features and K learnable offsets, with shared MLPs decoding attributes from anchor features, viewing direction, and distance. This decouples neural queries from the rendered Gaussian count. HAC exploits hash-assisted context for anchor compression (Chen et al., 2024), while dynamic variants extend anchor-based representations to space and time (Cho et al., 2026; Huang et al., 2025; Li et al., 2025; Chen et al., 2025a). NGMS instead stores complete, directly renderable diffuse Gaussians and shares residual geometry, view-dependent SH appearance, and optional temporal variation, replacing per-anchor features and offset tables with explicit Gaussian references.

Gaussian compression. Gaussian compression reduces encoded model size through pruning, quantization, spatial ordering, entropy coding, and rate-distortion optimization (Lee et al., 2024a; Fan et al., 2024; Morgenstern et al., 2024; Bagdasarian et al., 2025; Lee et al., 2025b;a). NGMS complements these techniques by reducing the learning parameters itself: N complete bases and a shared field generate NK rendered components.

3 Method

In Neural Gaussian Mixture Splatting (NGMS), explicit Gaussians provide the scene structure, while a shared neural field supplies view-dependent detail and temporal variation. Each base stores geometry and diffuse appearance; the field predicts K residuals that turn it into a mixture of renderable Gaussians. This complementary design supports compact static and dynamic reconstruction within the standard splatting pipeline (Figure 2).

3.1 Gaussian Bases and Rendering

An explicit Gaussian provides a compact reference for both geometry and appearance. We retain this structure in the base representation,

$$\mathcal{G} = \{g_i = (\boldsymbol{\mu}_i, \boldsymbol{q}_i, \boldsymbol{s}_i, o_i, \boldsymbol{f}_i)\}_{i=1}^{N}, \tag{1}$$

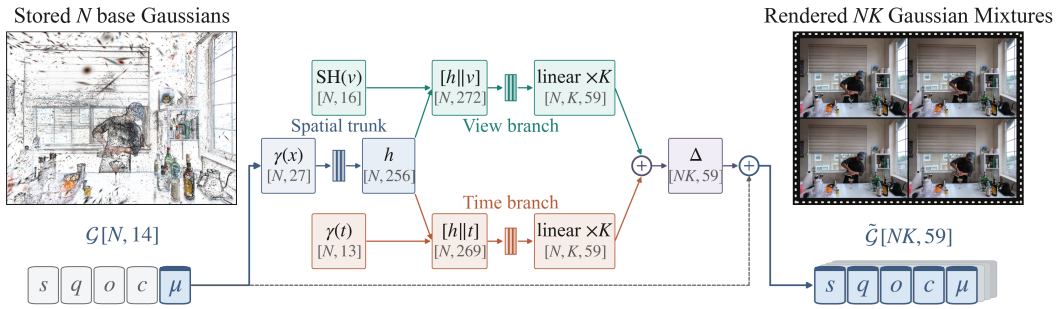


Figure 2: **NGMS pipeline.** A shared residual field expands N stored base Gaussians $\mathcal{G}$ into NK rendered Gaussians $\widetilde{\mathcal{G}}$. A spatial trunk encodes each base mean $\boldsymbol{\mu}$ into a descriptor $\boldsymbol{h}$. View and optional time branches predict K residuals that refine geometry, opacity, and appearance before standard Gaussian rasterization. All components of a base share one nonlinear field evaluation.

where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is the position, $\boldsymbol{q}_i \in \mathbb{R}^4$ a unit quaternion, $\boldsymbol{s}_i \in \mathbb{R}^3_+$ the scale, and $o_i \in (0,1)$ the opacity. Unlike the full SH appearance stored by each 3DGS primitive (Kerbl et al., 2023), $\boldsymbol{f}_i \in \mathbb{R}^{1\times 3}$ contains only the diffuse (DC) term. Each base thus remains directly renderable with just 14 scalars; the shared field supplies finer geometry and view-dependent appearance.

We optimize unconstrained rotation, log-scale, and opacity variables, with

$$\boldsymbol{q}_i = \frac{\bar{\boldsymbol{q}}_i}{\|\bar{\boldsymbol{q}}_i\|_2}, \qquad \boldsymbol{s}_i = \exp(\boldsymbol{\ell}_i), \qquad o_i = \mathrm{sigmoid}(\bar{o}_i). \tag{2}$$

Rotation and scale define the covariance $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top$, where $\boldsymbol{R} = \boldsymbol{R}(\boldsymbol{q})$ and $\boldsymbol{S} = \mathrm{diag}(\boldsymbol{s})$. Projection gives the image-plane mean $\boldsymbol{\mu}'_i$ and covariance $\boldsymbol{\Sigma}'_i$. At pixel $\boldsymbol{x}$, the opacity-weighted Gaussian contribution is

$$\sigma_i(\boldsymbol{x}) = o_i \exp\!\left(-\tfrac{1}{2} r_i(\boldsymbol{x})\right), \qquad r_i(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{\mu}'_i)^\top(\boldsymbol{\Sigma}'_i)^{-1}(\boldsymbol{x}-\boldsymbol{\mu}'_i). \tag{3}$$

For M primitives sorted front to back, standard alpha compositing yields

$$\boldsymbol{C}(\boldsymbol{x}) = \sum_{i=1}^{M} \boldsymbol{c}_i\, \sigma_i(\boldsymbol{x}) \prod_{j=1}^{i-1} (1-\sigma_j(\boldsymbol{x}))\,, \tag{4}$$

where $\boldsymbol{c}_i$ is the SH color evaluated for the current camera; the background term is omitted for brevity. The same formulation renders either the N bases or their NK expanded components.

3.2 Shared Residual Mixtures

The base Gaussians establish the scene structure; the residual field adds the detail needed for each view and time. Rather than querying a network separately for every rendered primitive, we share a spatial descriptor across the K components of each base. For a camera center $\boldsymbol{p}_{\mathrm{cam}}$, conditioning uses the base-to-camera direction expressed in the base's local frame:

$$\boldsymbol{v}_i = \frac{\boldsymbol{p}_{\mathrm{cam}} - \boldsymbol{\mu}_i}{\|\boldsymbol{p}_{\mathrm{cam}} - \boldsymbol{\mu}_i\|_2}, \qquad \boldsymbol{d}_i = \frac{\boldsymbol{R}(\boldsymbol{q}_i)^\top \boldsymbol{v}_i}{\|\boldsymbol{R}(\boldsymbol{q}_i)^\top \boldsymbol{v}_i\|_2}. \tag{5}$$

A spatial trunk encodes the base position, and separate branches combine this descriptor with the viewing direction and optional normalized time $t \in [0,1]$:

$$\begin{aligned} \boldsymbol{h}_i &= f^x_\theta(\gamma_x(\boldsymbol{\mu}_i))\,, \\ \boldsymbol{h}^v_i &= f^v_\theta([\boldsymbol{h}_i, \gamma_d(\boldsymbol{d}_i)])\,, \qquad \boldsymbol{h}^t_i = f^t_\theta([\boldsymbol{h}_i, \gamma_t(t)])\,. \end{aligned} \tag{6}$$

One linear head per active branch jointly predicts the K residual tuples:

$$\boldsymbol{\Delta}_{i,1:K} = H^v(\boldsymbol{h}^v_i) + \delta_{\mathrm{dyn}} H^t(\boldsymbol{h}^t_i), \qquad \boldsymbol{\Delta}_{i,k} = (\Delta\boldsymbol{\mu}_{i,k}, \Delta\boldsymbol{q}_{i,k}, \Delta\boldsymbol{s}_{i,k}, \Delta o_{i,k}, \Delta\boldsymbol{f}_{i,k}). \tag{7}$$

Here, Δo is an opacity-logit correction and $\Delta \boldsymbol{f} \in \mathbb{R}^{(L+1)^2 \times 3}$ contains all SH coefficients through degree L, including DC. Both branches predict geometry, opacity, and appearance, allowing the same residual construction to capture view-dependent detail and temporal deformation. Static scenes use $\delta_{\mathrm{dyn}} = 0$ and omit the time branch entirely; dynamic scenes use $\delta_{\mathrm{dyn}} = 1$.

Composing each predicted residual with its base gives a renderable mixture component:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{i,k} = \boldsymbol{\mu}_i + \Delta\boldsymbol{\mu}_{i,k}, \quad \tilde{\boldsymbol{s}}_{i,k} = \boldsymbol{s}_i + \Delta\boldsymbol{s}_{i,k}, \quad \tilde{\boldsymbol{q}}_{i,k} = \mathrm{normalize}(\boldsymbol{q}_i + \Delta\boldsymbol{q}_{i,k}), \\ \tilde{o}_{i,k} = \frac{\mathrm{sigmoid}(\bar{o}_i + \Delta o_{i,k})}{K}, \qquad \tilde{\boldsymbol{f}}_{i,k} = \mathrm{pad}_L(\boldsymbol{f}_i) + \Delta\boldsymbol{f}_{i,k}. \end{aligned} \tag{8}$$

Position corrections act in world coordinates, while the quaternion is renormalized after addition. Scale residuals are added to the activated base scales without a further positivity projection; the resulting factors enter the squared covariance construction above. Opacity is refined in logit space, and the $1/K$ factor scales component strength with mixture size. For appearance, pad_L retains the base DC term and fills higher-order coefficients with zeros before residual addition. The resulting view-conditioned SH features are evaluated by the rasterizer as usual. Thus, $\widetilde{\mathcal{G}} = \{\tilde{g}_{i,k}\}_{i=1,k=1}^{N,K}$ contains ordinary Gaussians and requires no change to the Gaussian rasterization or compositing equations. The input encodings and the network architecture are detailed in Appendix A.

3.3 Compactness and Computation

The key storage advantage is that rendered detail is generated, rather than independently parameterized. One network query produces K Gaussians in NGMS. For each base, the backbone (the spatial trunk and active conditioning branches) computes features that the linear heads turn into K residual tuples. Predicting these components separately would require K backbone evaluations instead of one. NGMS therefore queries only N base sites while still rendering all NK Gaussians.

The same sharing keeps storage compact. A degree-L Gaussian or residual tuple contains $D_g(L) = 11 + 3(L+1)^2$ scalars, giving 59 values at $L = 3$. NGMS stores only 14 scalars per base together with the shared field, rather than a separate attribute vector for every rendered component. For the same NK rendered primitives, the parameter counts are

$$P_{\mathrm{explicit}} = NKD_g(L), \qquad P_{\mathrm{NGMS}} = 14N + P_\theta(K), \tag{9}$$

where $P_\theta(K)$ counts the shared field parameters. Its backbone is independent of K, while each linear head maps W features to $KD_g(L)$ outputs. Including the head biases gives

$$P_\theta(K) = P_{\mathrm{backbone}} + B(W+1)KD_g(L), \tag{10}$$

where P_{backbone} is the backbone parameter count, W is the hidden width, and $B = 1$ or 2 heads are used for static or dynamic scenes.

In practice, W is much smaller than the number of splats NK that are required to reconstruct the scene: our network uses $W = 256$, while, for context, vanilla 3DGS uses 0.2-0.5 M Gaussians for synthetic objects and 1-5 M for captured scenes. Widening a shared head therefore costs far less than storing additional attributes for every rendered Gaussian.

Expanding each base into more components does not by itself reduce the number of stored bases. Our mixture-aware densification (Section 3.4) adapts base allocation to the expanded representation, allowing the count N to decrease as mixtures provide more detail. Section 4 evaluates this trade-off using complete compressed checkpoints.

3.4 Optimization

We learn the base Gaussians and residual field from posed images using the reconstruction objective

$$\mathcal{L}_{\mathrm{rec}} = (1 - \lambda_{\mathrm{SSIM}})\|\widehat{\boldsymbol{C}} - \boldsymbol{C}\|_1 + \lambda_{\mathrm{SSIM}} \left(1 - \mathrm{SSIM}(\widehat{\boldsymbol{C}}, \boldsymbol{C})\right), \tag{11}$$

where $\widehat{\boldsymbol{C}}$ and $\boldsymbol{C}$ denote the rendered and ground-truth images, and $\lambda_{\mathrm{SSIM}} = 0.2$.

Starting from base Gaussians. We begin with ordinary diffuse Gaussian splatting: during warm-up, the residual field is disabled and each base is rendered once with its own geometry, appearance, and opacity. Small random initialization of the position-head weights and biases breaks component symmetry, while all other residual output rows start at zero. Mixture expansion is introduced only when joint optimization begins.

Mixture-aware densification. To turn the additional rendering capacity into compact storage, we adapt base allocation to the detail already supplied by each mixture. The K components jointly inform whether their base needs further refinement. For each base, we pool the component screen-space gradients into a root-mean-square score:

$$\rho_i^{\mathrm{ss}} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left\|\frac{\partial \mathcal{L}_{\mathrm{rec}}}{\partial \boldsymbol{u}_{i,k}}\right\|_2^2}, \tag{12}$$

where $\boldsymbol{u}_{i,k} \in \mathbb{R}^2$ is the screen-space mean proxy for component (i, k). Pooling magnitudes avoids directional cancellation, while division by K prevents the score from growing merely because a mixture contains more components. At each refinement event, we sample this score for bases with at least one visible component, use it to guide base cloning and splitting, and prune low-opacity bases. Refinement thus responds to the reconstruction needs of the whole mixture, linking the rendered components to the stored base count. The ablation in Table 3 isolates the compactness and efficiency gains from this design. During base-only warm-up, the score reduces to the ordinary screen-space gradient magnitude. Training settings are provided in Appendix A.

4 Experiments

We evaluate Neural Gaussian Mixture Splatting on static and dynamic novel-view synthesis to assess how effectively shared residuals combine visual quality, compact storage, and real-time rendering.

4.1 Experimental Setup

Datasets. We use four benchmarks covering synthetic and real-world scenes in both static and dynamic settings. NeRF Synthetic (Mildenhall et al., 2020) evaluates fine geometric detail under controlled conditions, while Mip-NeRF 360 (Barron et al., 2022) tests unbounded real-world reconstruction. D-NeRF (Pumarola et al., 2021) provides synthetic sequences with diverse deformations, and Neural 3D Video (N3V) (Li et al., 2022) evaluates real-world dynamic scenes. Together, these benchmarks assess spatial detail, view-dependent appearance, and temporal consistency.

Baselines. Our baseline comparisons use 3DGS (Kerbl et al., 2023) and Scaffold-GS (Lu et al., 2024a) for static scenes, and 4DGS (Yang et al., 2024a) and 4D Scaffold-GS (Cho et al., 2026) for dynamic scenes. Appendix E adds broader PSNR and storage comparisons with published results.

Metrics. We assess reconstruction quality using PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). We report complete serialized model size in MB, training time, and rendering FPS to evaluate compactness and efficiency. For dynamic scenes, we additionally measure temporal coherence on N3V using tPSNR, tErr, and flow-warping residual.

Implementation details. NGMS is implemented in PyTorch using tiny-cuda-nn (Müller, 2021) for the residual field and the standard differentiable Gaussian rasterizer. Our experiments run on a single NVIDIA RTX A6000 GPU. Network architecture and optimization settings are provided in Appendix A. We downsample images by a factor of 2 on D-NeRF, N3V, and indoor Mip-NeRF 360, by 4 on outdoor Mip-NeRF 360, and use full resolution on NeRF Synthetic. Our comparisons strictly follow prior evaluation protocols and save only final-iteration checkpoints, without selecting or saving a best intermediate model. All reported NGMS checkpoints store Morton-ordered bases (FP32 positions, INT8 color/opacity/rotation, FP16 log-scales) and FP16 residual weights in one file. Our experiments use the same renderer after mixture expansion, so differences arise from the representation and optimization rather than a separate rendering backend.

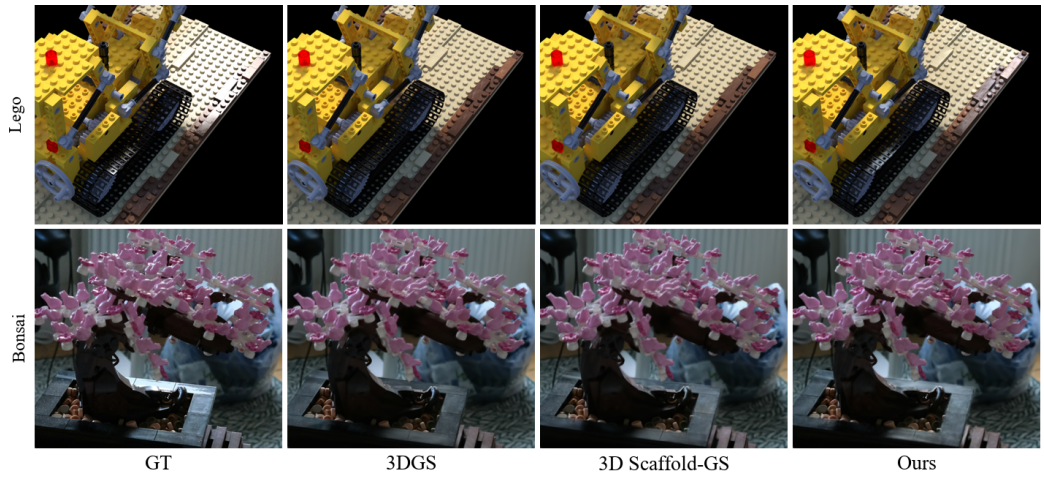


Figure 3: **Qualitative comparison on static-scene test views.**

Table 1: Per-scene comparison on the static benchmarks NeRF Synthetic (Mildenhall et al., 2020) and Mip-NeRF 360 (Barron et al., 2022). Train is in minutes; Mem is in MB.

Dataset	Scene	3DGS				Scaffold GS				**Ours**			
		PSNR↑	Train↓	FPS↑	Mem↓	PSNR↑	Train↓	FPS↑	Mem↓	PSNR↑	Train↓	FPS↑	Mem↓
NeRF Synthetic	chair	35.62	7.8	217.0	103.5	34.96	7.7	297.1	13.9	35.02	9.9	314.4	**2.3**
	drums	26.30	6.7	299.9	80.9	26.22	8.3	230.8	23.2	26.23	9.9	311.2	**2.4**
	ficus	35.54	4.0	502.0	39.8	35.16	7.0	343.8	12.9	35.83	7.0	527.4	**1.3**
	hotdog	38.18	6.4	368.2	41.9	37.84	8.3	341.6	8.4	37.86	8.6	302.3	**1.4**
	lego	36.14	7.0	286.9	74.5	35.41	8.1	254.4	18.2	35.96	11.6	248.8	**2.9**
	materials	30.52	4.5	565.6	33.9	30.58	7.8	262.2	21.2	30.43	8.4	440.6	**1.6**
	mic	36.88	5.0	338.7	41.4	36.71	7.3	334.4	10.3	36.02	7.3	422.1	**1.3**
	ship	31.64	6.6	288.2	58.5	31.11	10.8	184.0	30.0	31.27	9.1	401.9	**1.6**
	avg	33.85	6.0	358.3	59.3	33.50	8.2	281.0	17.3	33.58	9.0	371.1	**1.9**
Mip-NeRF 360	bicycle	25.62	35.5	61.8	1103.2	25.70	23.3	97.5	290.4	24.72	29.6	82.0	**7.6**
	flowers	21.93	24.2	113.4	652.0	21.76	22.3	127.3	213.7	21.38	33.1	85.1	**8.4**
	garden	27.87	33.1	78.7	863.1	27.89	24.8	92.2	198.4	27.30	43.5	62.5	**14.6**
	stump	26.91	29.2	81.3	1005.3	26.89	19.4	149.4	245.9	26.05	26.4	115.8	**5.8**
	treehill	22.86	26.9	94.8	739.6	23.39	22.2	125.4	228.5	23.23	30.7	87.2	**7.2**
	room	31.71	28.1	108.3	293.1	32.28	24.7	134.3	82.6	31.62	39.5	77.9	**5.3**
	counter	29.14	26.9	111.3	245.7	29.66	29.6	97.2	83.8	29.13	43.2	80.9	**6.7**
	kitchen	31.50	32.8	89.9	369.4	31.87	34.7	78.3	101.9	30.93	46.9	72.1	**8.3**
	bonsai	32.39	22.0	146.7	250.4	32.84	26.2	101.7	122.7	32.56	43.2	80.5	**8.1**
	avg	27.77	28.7	98.5	613.5	28.03	25.2	111.5	174.2	27.44	37.4	82.7	**8.0**

4.2 Main Results

Static scenes. NGMS preserves fine scene detail with a compact representation. Figure 3 shows representative test views from NeRF Synthetic and Mip-NeRF 360, including the repeated studs in `lego` and the fine branches and reflective surfaces in `bonsai`.

Table 1 shows that NGMS achieves the smallest serialized model on every static scene. On NeRF Synthetic, it reaches 33.58 dB PSNR with only 1.9 MB on average, compared with 33.85 dB at 59.3 MB for 3DGS and 33.50 dB at 17.3 MB for Scaffold-GS. On Mip-NeRF 360, NGMS uses 8.0 MB while remaining within 0.33 dB of 3DGS and 0.59 dB of Scaffold-GS. These results demonstrate substantial storage savings with a modest quality trade-off on unbounded real-world scenes. Per-scene SSIM/LPIPS and the NGMS storage breakdown are in Appendix B.

Dynamic scenes. NGMS extends this compact representation to dynamic scenes through time-conditioned residuals. Figure 4 shows clearer skeletal structure and character boundaries than 4DGS and 4D Scaffold-GS, whose reconstructions exhibit visible blur and artifacts in these views.

Table 2 compares NGMS with 4DGS and 4D Scaffold-GS on D-NeRF and N3V. On D-NeRF, NGMS improves average PSNR by 6.53 dB over 4DGS and 10.34 dB over 4D Scaffold-GS, achieving higher quality on every evaluated scene. On N3V, it reaches 31.37 dB, exceeding 4DGS by

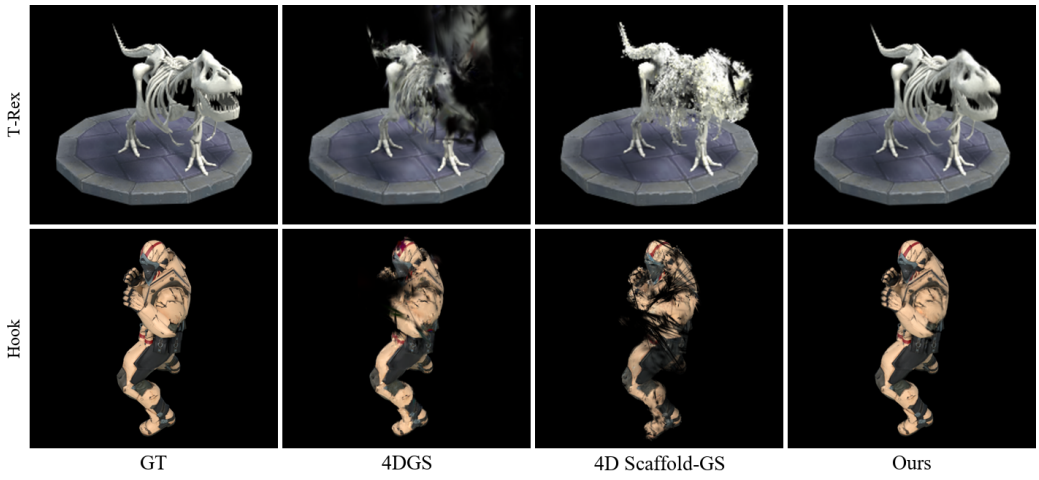


Figure 4: **Qualitative comparison on dynamic-scene test views.**

Table 2: Per-scene comparison on the dynamic benchmarks D-NeRF (Pumarola et al., 2021) and Neural 3D Video (Li et al., 2022). Train is in minutes; Mem is in MB.

Dataset	Scene	4DGS				4D Scaffold GS				**Ours**			
		PSNR↑	Train↓	FPS↑	Mem↓	PSNR↑	Train↓	FPS↑	Mem↓	PSNR↑	Train↓	FPS↑	Mem↓
D-NeRF	bouncingballs	33.36	8.8	236.2	129.5	30.76	8.7	197.4	25.0	43.20	7.5	322.1	**1.8**
	hellwarrior	34.84	8.2	287.3	153.5	30.06	9.6	183.5	35.7	41.08	6.0	322.2	**1.2**
	hook	31.97	10.2	230.9	233.9	28.07	9.6	167.8	37.0	36.83	7.4	277.0	**1.8**
	jumpingjacks	30.29	15.7	171.8	522.0	28.39	12.7	103.9	85.1	37.73	6.8	343.1	**1.6**
	mutant	38.65	7.2	257.3	100.6	33.66	7.5	277.9	8.0	42.48	6.2	340.1	**1.3**
	standup	38.97	8.4	252.5	117.3	32.32	8.0	192.3	14.8	44.55	6.2	429.7	**1.3**
	trex	29.77	25.4	110.7	1018.8	27.96	13.3	101.4	83.6	37.71	6.8	266.3	**1.6**
	avg	33.98	12.0	221.0	325.1	30.17	9.9	174.9	41.3	40.51	6.7	328.6	**1.5**
Neural 3D Video	coffee_martini	28.26	131.2	44.4	3781.7	28.67	43.2	47.6	150.4	29.45	27.2	45.4	**9.8**
	cook_spinach	33.09	87.2	63.0	2517.9	32.91	44.6	39.7	132.9	32.24	24.6	46.9	**9.8**
	cut_roasted_beef	31.34	86.3	61.9	2546.5	32.80	52.0	32.8	198.8	32.19	24.2	45.3	**9.5**
	flame_salmon	28.82	135.8	43.4	3901.6	29.01	45.6	42.0	160.9	28.97	28.1	37.6	**11.6**
	flame_steak	28.98	106.4	52.1	3241.4	33.93	49.6	40.6	153.3	32.19	24.6	47.7	**9.5**
	sear_steak	31.88	82.7	70.3	2161.9	33.59	44.1	46.2	119.2	33.16	24.6	44.6	**9.8**
	avg	30.40	104.9	55.9	3025.2	31.82	46.5	41.5	152.6	31.37	25.6	44.6	**10.0**

0.97 dB and remaining within 0.45 dB of 4D Scaffold-GS while using only 10.0 MB. NGMS also achieves the highest average tPSNR on N3V (49.0 versus 46.6 for 4D Scaffold-GS and 43.3 for 4DGS) and the lowest average flow-warping residual (Appendix C), supporting its ability to capture coherent temporal variation.

4.3 Compactness and Efficiency

NGMS achieves substantial storage reductions across all four benchmarks (Tables 1 and 2). On NeRF Synthetic, its models are $31\times$ smaller than 3DGS and $9.1\times$ smaller than Scaffold-GS; on Mip-NeRF 360, these reductions reach $77\times$ and $22\times$. On D-NeRF, NGMS combines higher reconstruction quality with $217\times$ and $28\times$ smaller models than 4DGS and 4D Scaffold-GS, respectively. On N3V, the corresponding reductions are $303\times$ and $15\times$.

These savings combine shared residual generation with compact encoding of both the base Gaussians and residual field.

NGMS also trains faster than both dynamic baselines, taking 6.7 minutes on D-NeRF and 25.6 minutes on N3V on average (Table 2). Querying the nonlinear trunk once per base and generating all K components with a linear head keeps neural evaluation efficient. NGMS achieves the highest average rendering speed among the compared methods on NeRF Synthetic and D-NeRF (371 and 329 FPS), and maintains real-time rendering on Mip-NeRF 360 and N3V (83 and 45 FPS). Static training takes longer than the baselines on average, reflecting the cost of learning the shared field alongside the base Gaussians.

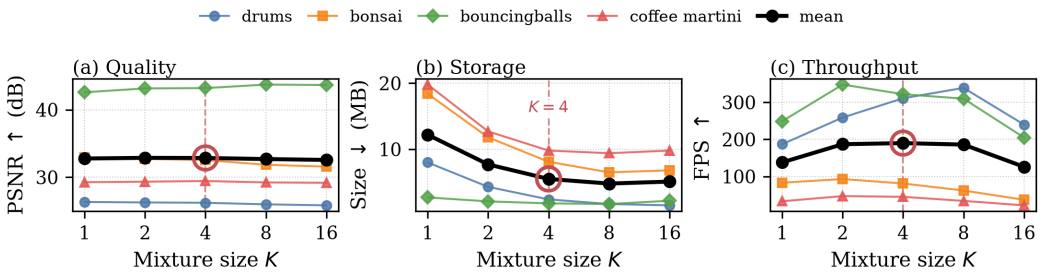


Figure 5: **Mixture-size sweep.** Per-scene curves and the four-scene mean versus K.

Table 3: **Ablation of the NGMS core design.** Mem: complete checkpoint size (MB); FPS: end-to-end rendering. Mean averages four scenes; shading marks the default (K=4). On static scenes, *No time branch* equals Full, and *No view branch* equals *No residual*.

	drums			bonsai			bouncingballs			coffee_martini			Mean		
Variant	PSNR↑	Mem↓	FPS↑	PSNR↑	Mem↓	FPS↑	PSNR↑	Mem↓	FPS↑	PSNR↑	Mem↓	FPS↑	PSNR↑	Mem↓	FPS↑
No time branch	26.23	2.4	311	32.56	8.1	81	23.20	1.6	448	27.00	9.6	73	27.25	5.4	228
No view branch	25.39	6.3	957	31.13	17.5	361	38.92	1.3	577	28.43	9.6	60	30.97	8.7	489
No residual	25.39	6.3	957	31.13	17.5	361	23.14	3.5	736	26.65	19.6	294	26.58	11.7	587
No mixture-aware densification	26.26	8.5	99	32.86	19.2	46	42.23	2.7	211	29.24	19.5	25	32.65	12.5	95
Full (K=4)	26.23	2.4	311	32.56	8.1	81	43.20	1.8	322	29.45	9.8	45	32.86	5.5	190

4.4 Ablation Studies

We evaluate mixture sizes $K \in \{1, 2, 4, 8, 16\}$ on one representative scene from each benchmark. Figure 5 shows that local expansion improves compactness while preserving quality: increasing K from 1 to 4 reduces mean storage from 12.2 MB to 5.5 MB, with mean PSNR increasing from 32.78 to 32.86 dB and rendering speed reaching its peak at 190 FPS. This balance motivates our default K=4. Mean storage falls further to 4.8 MB at K=8, but rises to 5.1 MB at K=16, while mean PSNR drops from 32.70 to 32.57 dB and speed from 186 to 126 FPS. This saturation is consistent with diminishing base-storage savings relative to growing shared heads (Section 3.3). The practical gain thus depends on preserving reconstruction quality, not arbitrarily increasing K. Quantitative results appear in Table 7 (Appendix D).

Table 3 evaluates the contribution of each component, with all variants retrained on the same four scenes. Removing time conditioning reduces PSNR on `bouncingballs` from 43.20 to 23.20 dB, demonstrating its role in modeling deformation. Removing view conditioning lowers mean PSNR by 1.89 dB, while removing all residual corrections lowers it by 6.28 dB.

Mixture-aware densification is important for translating mixture expansion into compact reconstruction. With $K = 4$ in both variants, it reduces mean storage from 12.5 to 5.5 MB and doubles rendering speed from 95 to 190 FPS, while improving mean PSNR from 32.65 to 32.86 dB. The same mixture capacity therefore gives substantially different storage and speed depending on how the bases are allocated and K-way expansion alone does not explain the gains.

5 Conclusion

We presented Neural Gaussian Mixture Splatting (NGMS), a unified framework that generates dense scene detail from compact Gaussian bases. Shared residual mixtures capture fine geometry, view-dependent appearance, and temporal deformation without independently storing every rendered primitive or its higher-order SH coefficients. Mixture-aware densification translates this capacity into storage savings by allocating bases according to the needs of each expanded mixture. The Gaussian-native design preserves standard rasterization while allowing Gaussian and network compression to work together. Across four benchmarks, NGMS achieves the smallest evaluated models with competitive static quality and real-time rendering. These results demonstrate that dense rendered detail need not imply dense parameter storage, making shared residual mixtures a practical foundation for compact static and dynamic Gaussian reconstruction.

AI USE STATEMENT

Generative AI tools assisted with drafting and polishing the manuscript, including organization, clarity, and consistency checks against the implementation and reported tables. Their use was limited to manuscript preparation; they were not used to develop the method, implement the models, conduct experiments, or generate experimental data. The authors reviewed the AI-assisted revisions and take responsibility for the final manuscript, including all claims and reported results.

ETHICS STATEMENT

This work studies compact static and dynamic scene reconstruction using four existing benchmarks, without new participant recruitment or data collection. As with other novel-view synthesis methods, reconstructed scenes may expose private information or be used to create misleading visual content. Applications should respect privacy, consent, and dataset terms, and distinguish synthesized views from captured observations.

REPRODUCIBILITY STATEMENT

Section 3 describes the representation, residual composition, training objective, and mixture-aware densification. Section 4 specifies the datasets, image preprocessing, evaluation protocol, hardware, and checkpoint storage. Appendix A provides the network architecture, initialization, and optimization settings, while Appendices B-D report per-scene results, storage breakdowns, temporal metrics, and mixture-size ablations. In addition, we will provide the code and data for the experiments in this paper after the paper acceptance.

REFERENCES

Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods. *CGF*, 44(2):e70078, 2025.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *ICCV*, 2021.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, pp. 5470–5479, 2022.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*, 2023.

Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. In *CVPR*, 2023.

Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, and Changil Kim. Textured Gaussians for Enhanced 3D Scene Appearance Modeling. In *CVPR*, pp. 8964–8974, 2025.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *ECCV*, pp. 333–350, 2022.

Jianing Chen, Zehao Li, Yujun Cai, Hao Jiang, Chengxuan Qian, Juyuan Kang, Shuqin Gao, Honglong Zhao, Tianlu Mao, and Yucheng Zhang. HAIF-GS: Hierarchical and Induced Flow-Guided Gaussian Splatting for Dynamic Scene. In *NeurIPS*, 2025a.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In *ECCV*, 2024.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC++: Towards 100X Compression of 3D Gaussian Splatting. *IEEE TPAMI*, 47(11):10210–10226, 2025b.

Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, and Seon Joo Kim. 4D Scaffold Gaussian Splatting with Dynamic-Aware Anchor Growing for Efficient and High-Fidelity Dynamic Scene Reconstruction. In *AAAI*, pp. 3363–3371, 2026.

Stavros Diolatzis, Tobias Zirr, Alexander Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. N-Dimensional Gaussians for Fitting of High Dimensional Functions. In *SIGGRAPH*, 2024.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *NeurIPS*, 2024.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, Xiuhong Li, Ninghui Sun, Xingcheng Zhang, and Bo Dai. FlashGS: Efficient 3D Gaussian Splatting for Large-scale and High-resolution Rendering. In *CVPR*, 2025.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *CVPR*, pp. 12479–12488, 2023.

Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 6DGS: Enhanced Direction-Aware Gaussian Splatting for Volumetric Rendering. In *ICLR*, 2025a.

Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 7DGS: Unified Spatial-Temporal-Angular Gaussian Splatting. In *ICCV*, 2025b.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. EAGLES: Efficient Accelerated 3D Gaussians with Lightweight EncodingS. In *ECCV*, pp. 54–71, 2024.

Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-Aware 3D Gaussian Splatting for Efficient Dynamic Scene Reconstruction. *IEEE TCSVT*, 35(4):3119–3133, 2025.

Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized Exponential Splatting for Efficient Radiance Field Rendering. In *CVPR*, 2024.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking Neural Radiance Fields for Real-Time View Synthesis. In *ICCV*, pp. 5875–5884, 2021.

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. In *CVPR*, 2025.

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Daniel Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. In *3DV*, 2026.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH*, 2024a.

He Huang, Qi Yang, Mufan Liu, Yiling Xu, and Zhu Li. ADC-GS: Anchor-Driven Deformable and Compressed Gaussian Splatting for Dynamic Scene Reconstruction. In *IJCAI*, pp. 1179–1187, 2025.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. In *CVPR*, pp. 4220–4230, 2024b.

Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A Compact Dynamic 3D Gaussian Representation for Real-Time Dynamic View Synthesis. In *ECCV*, pp. 394–412, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4), 2023.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *NeurIPS*, 2024.

Sangwoon Kwak, Joonsoo Kim, Jun Young Jeong, Won-Sik Cheong, Jihyong Oh, and Munchurl Kim. MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting. In *CVPR*, pp. 11338–11348, 2025.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *CVPR*, 2024a.

Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized Minimal 3D Gaussian Splatting. In *NeurIPS*, 2025a.

Junoh Lee, Changyeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully Explicit Dynamic Gaussian Splatting. In *NeurIPS*, pp. 5384–5409, 2024b.

Soonbin Lee, Fangwen Shu, Yago Sanchez, Thomas Schierl, and Cornelius Hellge. Compression of 3D Gaussian Splatting with Optimized Feature Planes and Standard Video Codecs. In *ICCV*, pp. 25496–25505, 2025b.

Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3D Video Synthesis from Multi-View Video. In *CVPR*, pp. 5521–5531, 2022.

Yilong Li, Bo Pang, Yisong Chen, and Guoping Wang. Anchored 4D Gaussian Splatting for Dynamic Novel View Synthesis. In *SIGGRAPH Asia*, 2025.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *CVPR*, pp. 8508–8520, 2024.

Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. In *CVPR*, pp. 21136–21145, 2024.

Mufan Liu, Qi Yang, He Huang, Wenjie Huang, Zhenlong Yuan, Zhu Li, Yiling Xu, and Yunfeng Guan. Light4GS: Lightweight Compact 4D Gaussian Splatting Generation via Context Model. *IEEE TCSVT*, 36(6):7992–8005, 2026a.

Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable Beta Splatting. In *SIGGRAPH*, 2025.

Rong Liu, Zhongpai Gao, Benjamin Planche, Meida Chen, Van Nguyen Nguyen, Meng Zheng, Anwesa Choudhuri, Terrence Chen, Yue Wang, Andrew Feng, and Ziyan Wu. Universal Beta Splatting. In *ICLR*, 2026b.

Ange Lou, Benjamin Planche, Zhongpai Gao, Yamin Li, Tianyu Luan, Hao Ding, Terrence Chen, Jack Noble, and Ziyan Wu. DaReNeRF: Direction-aware Representation for Dynamic Scenes. In *CVPR*, pp. 5031–5042, 2024.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *CVPR*, pp. 20654–20664, 2024a.

Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D Geometry-Aware Deformable Gaussian Splatting for Dynamic View Synthesis. In *CVPR*, pp. 8900–8910, 2024b.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *3DV*, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *ECCV*, 2024.

Thomas Müller. tiny-cuda-nn. Open-source software, 2021. URL https://github.com/NVlabs/tiny-cuda-nn.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4):102:1–102:15, 2022.

K. L. Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization. In *ECCV*, pp. 330–349, 2024.

Niantic Labs. SPZ: A Compact File Format for 3D Gaussian Splats. Open-source software, 2024. URL https://github.com/nianticlabs/spz.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *CVPR*, pp. 10349–10358, 2024.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields. In *ICCV*, pp. 5865–5874, 2021a.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM TOG*, 40(6):238:1–238:12, 2021b.

Kyungdae Park and Chae Eun Rhee. CC-4DGS: Computational Deformation and Point-Cloud Compression for Storage-Efficient Dynamic Gaussian Splatting. *IEEE TVCG*, pp. 1–16, 2026. Early access.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, pp. 10318–10327, 2021.

Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising Densification in Gaussian Splatting. In *ECCV*, pp. 347–362, 2024.

Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware Gaussian Compression for Fast and High-quality Rendering. In *ICLR*, 2025.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction. In *CVPR*, pp. 5459–5469, 2022.

Boyuan Tian, Qizhe Gao, Siran Xianyu, Xiaotong Cui, and Minjia Zhang. FlexGaussian: Flexible and Cost-Effective Training-Free Compression for 3D Gaussian Splatting. In *ACM MM*, pp. 7287–7296, 2025.

Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C. Kot, and Bihan Wen. ContextGS: Compact 3D Gaussian Splatting with Anchor Level Context Model. In *NeurIPS*, 2024.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE TIP*, 13(4):600–612, 2004.

Zipeng Wang and Dan Xu. HyRF: Hybrid Radiance Fields for Memory-efficient and High-quality Novel View Synthesis. In *NeurIPS*, 2025.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *CVPR*, pp. 20310–20320, 2024.

Jiahao Wu, Rui Peng, Jianbo Jiao, Jiayu Yang, Luyang Tang, Kaiqiang Xiong, Jie Liang, Jinbo Yan, Runling Liu, and Ronggang Wang. LocalDyGS: Multi-view Global Dynamic Scene Modeling via Adaptive Local Implicit Feature Decoupling. In *ICCV*, pp. 9519–9529, 2025.

Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering. In *CVPR*, 2024.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *ICLR*, 2024a.

Ziyi Yang, Xinyu Gao, Yang-Tian Sun, Yi-Hua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting. In *NeurIPS*, 2024b.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. In *CVPR*, 2024c.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-Time Rendering of Neural Radiance Fields. In *ICCV*, pp. 5752–5761, 2021.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *CVPR*, pp. 19447–19456, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018.

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density Control with Pixel-Aware Gradient for 3D Gaussian Splatting. In *ECCV*, pp. 326–342, 2024.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA Volume Splatting. In *IEEE VIS*, pp. 29–36, 2001a.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface Splatting. In *SIGGRAPH*, pp. 371–378, 2001b.