

Toward Efficient Static and Dynamic Gaussian Splatting

□ 윤석현, 오지형 / 중앙대학교

Abstract

3D Gaussian Splatting (3DGS) is a powerful explicit representation for high-quality, real-time 3D reconstruction and novel view synthesis, but the memory and computational cost of storing and rendering millions of Gaussians hinders deployment, and this worsens in dynamic 4D scenes. To reduce such redundancy while preserving quality, research on Efficient Gaussian Splatting has advanced rapidly. This survey categorizes efficiency-oriented approaches for both static (3D) and dynamic (4D) Gaussian Splatting into two paradigms, Parameter Compression and Restructuring Compression, reviews the key ideas in each, and overviews widely used datasets and evaluation metrics.

I. Introduction

3D scene representation and photorealistic novel view synthesis underpin applications such as virtual reality (VR), augmented reality (AR), media generation, autonomous driving, and large-scale visualization. The field advanced markedly with Neural Radiance Fields (NeRF) [1], which optimizes an implicit representation via differentiable volume rendering to reconstruct high-quality 3D from 2D

images, but is costly to train and render, as each pixel needs many ray samples.

Unlike NeRF's implicit weights, classical explicit representations such as point clouds and meshes model scenes as primitive collections. 3D Gaussian Splatting (3DGS) [2] revives this explicit paradigm, modeling a scene as anisotropic 3D Gaussians with position, scale, rotation, opacity, and view-dependent spherical-harmonics color. Learned directly from multi-view images, 3DGS attains high-quality

reconstruction and real-time rendering, infeasible for NeRF.

Dynamic 3D Gaussian Splatting (4D Gaussian Splatting) further extends static 3DGS with a temporal dimension to reconstruct non-rigidly moving scenes from monocular or multi-view video, enabling dynamic scene editing, motion capture, and realistic simulation for robotics and autonomous driving.

However, deployment is hindered by heavy memory and computation: static scenes often hold millions of Gaussians, and 4D reconstruction is costlier still, as each Gaussian also encodes temporal information.

To address these bottlenecks, a new research direction, termed **Efficient Gaussian Splatting**, has emerged to reduce the redundancy of Gaussian primitives while preserving rendering quality. This survey categorizes the existing literature on Efficient Gaussian Splatting into the following two directions:

- **Parameter Compression:** Directly compresses Gaussian attributes through techniques such as pruning, quantization, and entropy coding.
- **Restructuring Compression:** Compresses Gaussian representations by redesigning the structure of the original Gaussian Splatting framework through hierarchical anchors, neural network integration, and geometry-aware representations.

Applying this taxonomy to both static 3DGS and 4D Gaussian Splatting (4DGS), this survey reviews representative methods and formulations (Sections 3 and 4) and summarizes widely used datasets and evaluation metrics (Section 5), aiming to provide a unified foundation for scalable and efficient Gaussian Splatting.

II. Preliminary

1. Static 3D Gaussian Splatting (3DGS)

3DGS represents a scene as anisotropic 3D Gaussians projected directly onto the 2D screen. Unlike NeRF, it needs no neural network and renders in real time via a CUDA-based tile rasterizer.

Initialization and Attributes. Structure-from-Motion (SfM) tools such as COLMAP recover a sparse point cloud and camera poses from multi-view images, and each point initializes a 3D Gaussian. Each Gaussian G_i has position $\mu_i \in \mathbb{R}^3$, scale $\mathbf{S}_i \in \mathbb{R}^3$, rotation $\mathbf{r}_i \in \mathbb{R}^4$ (a unit quaternion, $\|\mathbf{r}_i\| = 1$), view-dependent spherical-harmonics (SH) coefficients $\mathbf{h}_i$, and opacity $\sigma_i \in [0, 1]$, so the set of N Gaussians is $\mathcal{G} = \{G_i = (\mu_i, \mathbf{S}_i, \mathbf{r}_i, \mathbf{h}_i, \sigma_i)\}_{i=1}^N$.

Gaussian Formulation. A Gaussian centered at μ_i is defined as

$$G_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^\top \Sigma_i^{-1}(\mathbf{x} - \mu_i)\right), \quad (1)$$

where its covariance is constructed from the rotation and scale matrices and is given by

$$\Sigma_i = \mathbf{R}_i \mathbf{S}_i \mathbf{S}_i^\top \mathbf{R}_i^\top, \quad (2)$$

where $\mathbf{R}_i$ and $\mathbf{S}_i$ are the rotation and scale matrices derived from $\mathbf{r}_i$ and $\mathbf{S}_i$, respectively.

Projection and Tile Rasterization. Via the world-to-camera transform $\mathbf{W}$ and the Jacobian $\mathbf{J}$ of the affine approximation to the camera-to-screen projection, the 3D covariance is projected into a 2D covariance, formulated as

$$\Sigma_i' = \mathbf{J} \mathbf{W} \Sigma_i \mathbf{W}^\top \mathbf{J}^\top, \quad (3)$$

so each Gaussian becomes a 2D ellipse on screen. The image is split into 16×16 tiles, and at each pixel $\mathbf{p}$ the alpha value is computed as

$$\alpha_i(\mathbf{p}) = \sigma_i \cdot g_i(\mathbf{p}), \quad (4)$$

where $g_i(\mathbf{p})$ is the projected 2D Gaussian evaluated at pixel $\mathbf{p}$. The final color is then obtained by alpha blending as follows:

$$\mathcal{C}(\mathbf{p}) = \sum_i \mathbf{c}_i \alpha_i(\mathbf{p}) \prod_{j < i} (1 - \alpha_j(\mathbf{p})). \quad (5)$$

Here, $\mathbf{c}_i$ is the view-dependent color of the i -th Gaussian decoded from its SH coefficients. The rasterizer is fully differentiable, so gradients flow to all Gaussian parameters.

Optimization and Adaptive Density Control. Train-

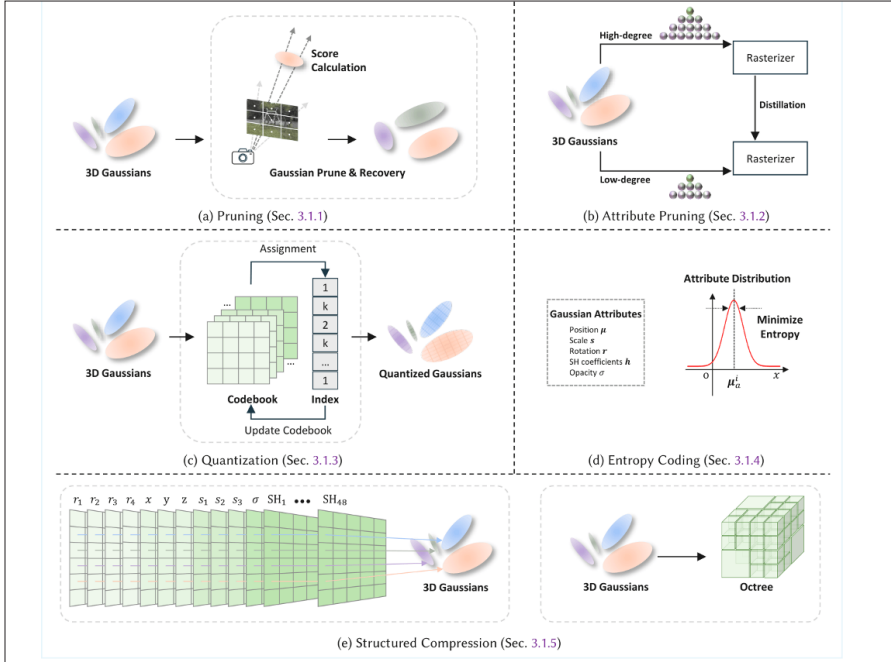
ing minimizes, between the rendered image I_{pred} and ground truth I_{gt} , the rendering loss defined as

$$L = (1 - \lambda)L_1 + \lambda(1 - \text{SSIM}), \quad (6)$$

where L_1 is the pixel-wise ℓ_1 loss and λ balances the L_1 and SSIM terms. As sparse points underfit the scene, Gaussians are cloned and split where the screen-space position gradient exceeds τ_{pos} , while near-transparent Gaussians are pruned to bound their number.

III. Efficient Static Gaussian Splatting

Static 3DGS needs many Gaussians for high fidelity.



<Fig. 1> Overview of Parameter Compression strategies for static 3DGS

Large scenes can reach millions of Gaussians, requiring gigabytes of storage and hindering deployment on resource-constrained devices. Efficient static 3DGS falls into two directions,

1. Parameter Compression

Parameter compression reduces storage and memory without modifying the 3DGS architecture and applies even to trained models. We group existing methods into five strategies, which are summarized in Fig. 1.

- 1) **Pruning.** Pruning removes redundant or low-contribution Gaussians. **Learnable mask-based pruning** (Lee et al. [3]) jointly considers scale and opacity via a learnable mask m_n , yielding a binary mask M_n regularized by the masking loss defined as

$$L_m = \frac{1}{N} \sum_n \text{Sig}(m_n). \quad (7)$$

Here, $\text{Sig}(\cdot)$ is the sigmoid function and N is the number of Gaussians.

Importance-score-based methods rank Gaussians by global importance (the GS score of Light-Gaussian [4]), pixel influence (EAGLES [5]), or local discriminability (OMG [6]). **Gradient-based** methods (PUP 3DGS [7], Speedy-Splat [8], ELMGS [9]) combine Hessian sensitivity, gradients, and opacity.

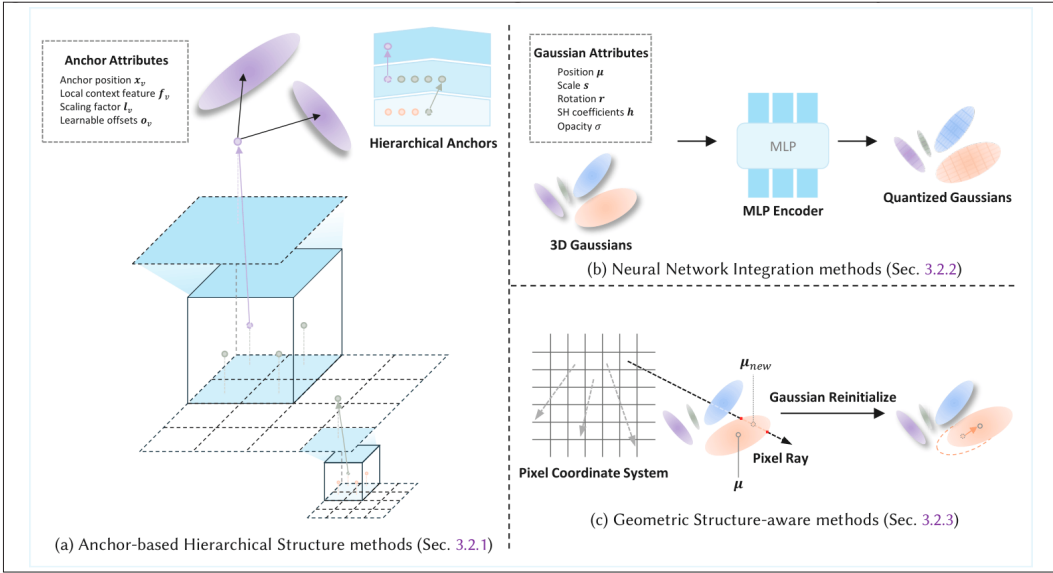
- 2) **Attribute Pruning.** This removes or compresses attribute components with limited impact on fidelity: LightGaussian distills SH coefficients to a

lower dimension, and RDO-Gaussian [10] masks specific attributes.

- 3) **Quantization.** Quantization lowers attribute bit precision. Niedermayr et al. [11] use sensitivity-based k -means and Lee et al. use residual vector quantization (R-VQ), while RDO-Gaussian selects codewords by joint rate-distortion. SizeGS [12] applies hierarchical mixed-precision quantization (H-MPQ) from file-size estimation, and Flex-Gaussian [13] enables training-free compression.
- 4) **Entropy Coding.** This exploits statistical redundancy in quantized attributes. Niedermayr et al. use DEFLATE for codebook indices and RDO-Gaussian uses arithmetic coding. HAC [14] models anchor attributes as Gaussians to estimate their probabilities for entropy coding, and FCGS [15] compresses in one forward pass without per-scene optimization.
- 5) **Structured Compression.** These methods exploit the spatial adjacency of unstructured Gaussians. PLAS [16] sorts attributes into a 2D grid. Octree-GS [17] selects the level of detail (LoD) via an octree. MesonGS [18] and HGSC [19] voxelize positions and hierarchically compress neighboring voxels. GoDe [20] builds a model-agnostic multi-level hierarchy without retraining.

2. Restructuring Compression

Restructuring compression modifies the 3DGS architecture itself for more efficient representations. We group these methods into three strategies, which



<Fig. 2> Overview of Restructuring Compression strategies for static 3DGS

are summarized in Fig. 2.

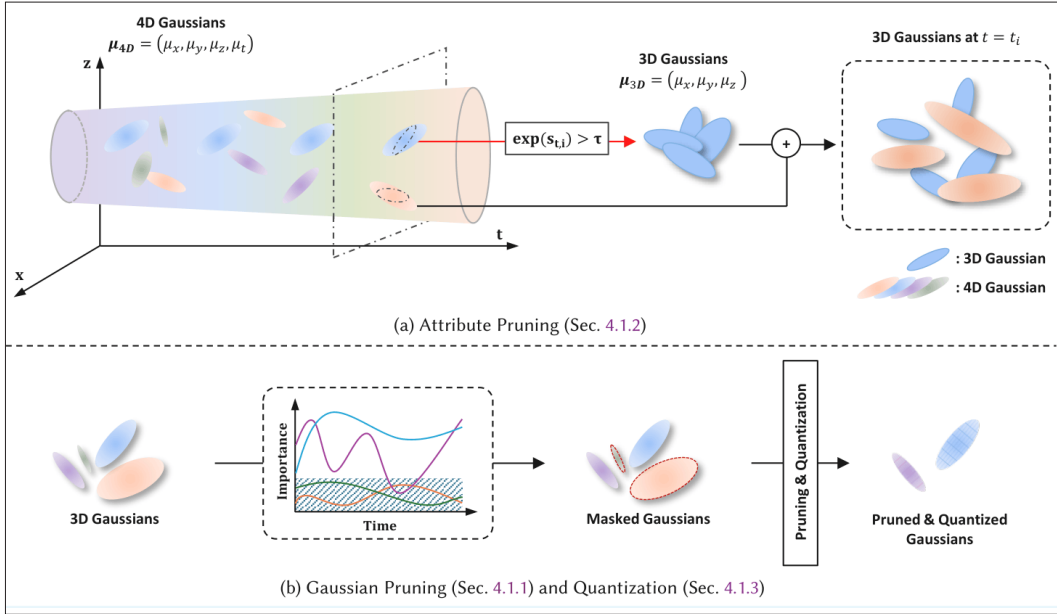
1) **Anchor-Based Hierarchical Structures.** Sparse anchors from voxelization serve as structural primitives. Scaffold-GS [21] gives each anchor local features, scales, and learnable offsets to spawn k neural Gaussians. HAC uses hash-grid features as context for entropy coding. ContextGS [22] predicts multi-level anchors autoregressively. CAT-3DGS [23], PCGS [24], TC-GS [25], and SHTC [26] further exploit spatial and channel-wise correlations.

2) **Neural Network Integration.** Neural networks replace Gaussian attributes via their generative capacity. EAGLES decodes quantized latents with an MLP. NeuralGS [27] assigns a small MLP to each cluster of similar attributes. Liu et al. [28] use a Mixture of Priors (MoP) with coarse-to-fine quantization (C2FQ).

3) **Geometry-Aware Structures.** Geometric priors are built into the representation. SAGS [29] uses curvature-aware densification with a GNN encoder. Mini-Splatting [30] rebuilds the spatial distribution via blur partitioning and depth reinitialization. 3D-HGS [31] uses a 3D half-Gaussian kernel with per-half opacities to capture discontinuities.

IV. Efficient Dynamic Gaussian Splatting

Dynamic scenes add non-rigid motion and time-varying geometry and appearance, so compact temporal representation is a central challenge as storage grows with motion. Following the static taxonomy, we analyze 4DGS by parameter and restructuring compression.



<Fig. 3> Overview of Parameter Compression strategies for 4DGS

1. Parameter Compression

These methods operate per Gaussian or attribute and integrate easily into existing pipelines. The strategies are summarized in Fig. 3.

- 1) **Gaussian Pruning.** Since Gaussian importance varies over time, TC3DGS [32] augments the static masking loss in Eq. (7) with a temporal mask-consistency term for stable pruning, defined as

$$\mathcal{L}_{mc} = \sum_n |m_{n,t} - \text{sg}(m_{n,t-1})|. \quad (8)$$

Here, $m_{n,t}$ is the mask of the n -th Gaussian at time t and $\text{sg}(\cdot)$ is the stop-gradient operator. Speede3DGS [33] uses a temporal sensitivity score and 4DGS-1K [34] combines temporal and spatial scores. Ex4DGS [35] prunes Gaussians

whose back-tracked image-space error exceeds a global threshold, 3DGStream [36] adds Gaussians for emerging objects with adaptive population control, and Instant4D [37] keeps only voxel-grid centers.

- 2) **Attribute Pruning.** Hybrid 3D-4DGS [38] detects Gaussians whose temporal extent is large enough to be treated as static and drops their temporal attributes, converting 4D Gaussians into 3D Gaussians and cutting parameters.
- 3) **Quantization.** Vector quantization (VQ) groups similar parameters into codebooks indexed by reference. TC3DGS allocates bit precision adaptively from gradient-based sensitivity.
- 4) **Entropy-Based Methods.** HiFi4G [39] quantizes

inter-frame residuals and encodes them with RANS for real-time long-sequence compression. MEGA [40] adds a spatial-opacity entropy loss defined as

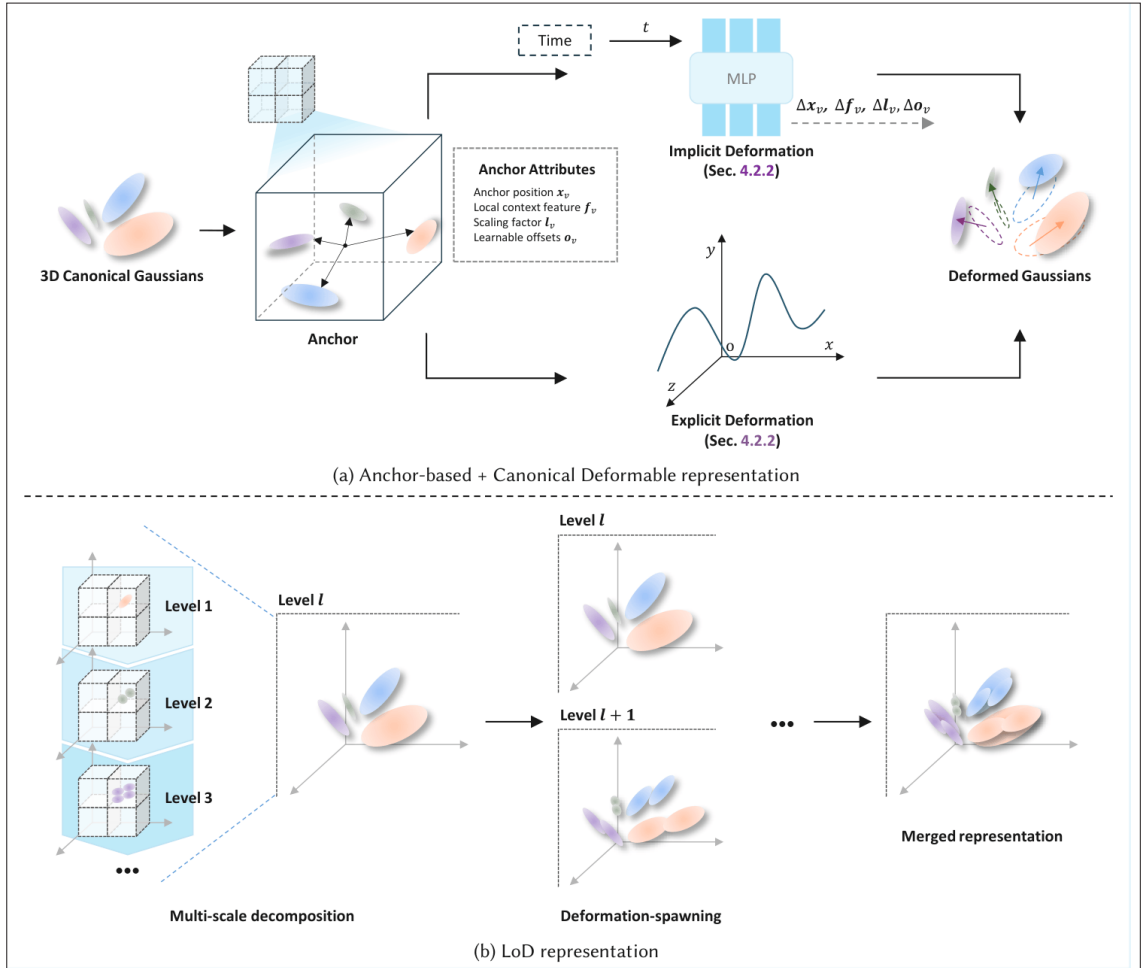
$$\mathcal{L}_{opa} = \frac{1}{N} \sum_j (-\sigma_j \log \sigma_j), \quad (9)$$

where σ_j is the opacity of the j -th Gaussian. This loss pushes opacities toward 0 or 1 and aggressively prunes near-zero ones.

2. Restructuring Compression

Restructuring compression reuses structures shared across frames. We group these methods into three approaches, which are summarized in Fig. 4.

1) **Anchor-Based Representations.** A few hierarchical anchors represent many Gaussians, densifying where motion is complex. MoScA [41] separates geometry and motion via a 4D motion



<Fig. 4> Overview of Restructuring Compression strategies for 4DGS

scaffold, EDGS [42] decomposes time-invariant and time-varying attributes with sparse anchor grids. MoDec-GS [43] introduces Global-Local Motion Decomposition (GLMD). HAIF-GS [44] uses hierarchical anchor densification with flow-guided deformation.

2) **Canonical Deformation Representations.** A canonical set of static Gaussians is mapped to time by a deformation field. **Implicit deformations** (Deformable 3DGS [45], the 4DGS of Wu et al. [46], DeformGS [47]) learn the field with MLPs or grids. **Explicit deformations** use predefined functions: Li et al. [48] use low-order polynomials, while Ex4DGS interpolates keyframes, with positions interpolated by a Cubic Hermite Spline as follows:

$$\mu(t) = \text{CHip}(p_n, m_n, p_{n+1}, m_{n+1}; t'), \quad (10)$$

where p_n and p_{n+1} are positions at two adjacent keyframes, m_n and m_{n+1} their tangents, and t' the normalized time between them. Rotations are interpolated by spherical linear interpolation, expressed as

$$q(t) = \text{Slerp}(r_n, r_{n+1}; t'), \quad (11)$$

where r_n and r_{n+1} are the rotations at the two keyframes. This keyframe interpolation avoids per-frame storage and shrinks the model.

3) **LoD Representations.** Detail is adjusted by viewpoint, distance, and importance. Scale-GS [49] partitions the scale space by recursive binary splitting for coarse-to-fine optimization, and 4DGCPro [50] separates motion into rigid transformation and residual deformation.

V. Datasets and Evaluation

Beyond rendering-quality metrics, efficient 3DGS requires efficiency metrics. While early work focused on visual quality (PSNR, SSIM, LPIPS), efficient methods target a quality-efficiency trade-off.

1. Datasets

Static Scene Datasets. As summarized in Table 1, representative datasets include Tanks and

<Table 1> Summary of representative static 3DGS datasets. In the #Scenes column, red and blue numbers indicate the numbers of outdoor and indoor scenes, respectively.

	TNT [51]	Deep Blending [52]	NeRF-Synthetic [1]	BungeeNeRF [53]	Mip-NeRF 360 [54]
Venue	ToG'17	ToG'18	ECCV'20	ECCV'22	CVPR'22
Type	Real	Real	Synthetic	Mixed	Real
Modality	Multi-view	Multi-view	Multi-view	Multi-view	Multi-view
#Views	100-400	12-418	300	220-463	100-330
#Scenes	14 (10+4)	19 (14+5)	8 (0+8)	12 (12+0)	9 (5+4)
Resolution	1920 × 1080	1228-2592 × 816-1944	800 × 800	N/A	4946 × 3286
Environment	Mixed	Mixed	Indoor	Outdoor	Mixed

<Table 2> Summary of representative 4DGS datasets. In the #Scenes column, red and blue numbers indicate the numbers of outdoor and indoor scenes, respectively.

	Technicolor [55]	D-NeRF [56]	HyperNeRF [57]	N3DV [58]	NeRF-DS [59]
Venue	CVPR'17	CVPR'21	ToG'21	CVPR'22	CVPR'23
Type	Real	Synthetic	Real	Real	Real
Modality	Multi-view	Multi-view	Monocular	Multi-view	Multi-view
#Views	16	100-200	1-2	18-21	2
#Scenes	11 (0+11)	8 (2+6)	7 (0+7)	6 (0+6)	8 (0+8)
#Frames	150-300	50-200	450-900	300	500
Resolution	2048 × 1088	800 × 800	1980 × 1080	2704 × 2028	480 × 270

Temples [51], Deep Blending [52], NeRF-Synthetic [1], BungeeNeRF [53], and Mip-NeRF 360 [54], spanning large-scale real, synthetic, multi-scale, and unbounded 360° scenes.

Dynamic Scene Datasets. As summarized in Table 2, common datasets include Technicolor [55], D-NeRF [56], HyperNeRF [57], Neural 3D Video [58] (N3DV), and NeRF-DS [59], covering light-field, synthetic, monocular topology-changing, multi-view, and specular dynamic scenes.

2. Evaluation Metrics

Fidelity is measured by **PSNR**, **SSIM**, and **LPIPS** (higher PSNR/SSIM and lower LPIPS are better). Efficiency is measured by **model size** (MB or number

of Gaussians) and **compression ratio** for compactness, and by **training time** and **inference FPS** for practicality, where smaller size, higher compression and FPS, and shorter training time are preferred.

VI. Conclusion

3DGS enables real-time rendering but at high memory cost. Under a unified taxonomy of **Parameter Compression** and **Restructuring Compression**, this survey reviews compact, efficient representations for static and dynamic scenes (Sections 3 and 4) and summarizes datasets and metrics (Section 5), aiming to serve as a foundation for scalable Gaussian Splatting.

참 고 문 헌

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” in ECCV, 2020.
- [2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” ACM ToG, 2023.
- [3] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, “Compact 3D Gaussian representation for radiance field,” in CVPR, 2024.
- [4] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, “LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS,” in NeurIPS, 2024.
- [5] S. Girish, K. Gupta, and A. Shrivastava, “EAGLES: Efficient accelerated 3D Gaussians with lightweight encodings,” in ECCV, 2024.
- [6] J. C. Lee, J. H. Ko, and E. Park, “Optimized minimal 3D Gaussian splatting,” in NeurIPS, 2025.
- [7] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, “PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting,” in CVPR, 2025.
- [8] A. Hanson, A. Tu, G. Lin, V. Singla, M. Zwicker, and T. Goldstein, “Speedy-Splat: Fast 3D Gaussian splatting with sparse pixels and sparse primitives,” in CVPR, 2025.
- [9] M. S. Ali, S. H. Bae, and E. Tartaglione, “ELMGS: Enhancing memory and computation scalability through compression for 3D Gaussian splatting,” in WACV, 2025.
- [10] H. Wang, H. Zhu, T. He, R. Feng, J. Deng, J. Bian, and Z. Chen, “End-to-end rate-distortion optimized 3D Gaussian representation,” in ECCV, 2024.
- [11] S. Niedermayr, J. Stumpfegger, and R. Westermann, “Compressed 3D Gaussian splatting for accelerated novel view synthesis,” in CVPR, 2024.
- [12] S. Xie, J. Liu, W. Zhang, S. Ge, S. Pan, C. Tang, Y. Bai, and Z. Wang, “SizeGS: Size-aware compression of 3D Gaussians with hierarchical mixed precision quantization,” in ACM MM, 2024.
- [13] B. Tian, Q. Gao, S. Xianyu, X. Cui, and M. Zhang, “FlexGaussian: Flexible and cost-effective training-free compression for 3D Gaussian splatting,” in ACM MM, 2025.
- [14] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, “HAC: Hash-grid assisted context for 3D Gaussian splatting compression,” in ECCV, 2024.
- [15] Y. Chen et al., “Fast feedforward 3D Gaussian splatting compression,” in ICLR, 2024.
- [16] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, “Compact 3D scene representation via self-organizing Gaussian grids,” in ECCV, 2024.
- [17] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai, “Octree-GS: Towards consistent real-time rendering with LoD-structured 3D Gaussians,” IEEE TPAMI, 2025.
- [18] S. Xie, W. Zhang, C. Tang, Y. Bai, R. Lu, S. Ge, and Z. Wang, “MesonGS: Post-training compression of 3D Gaussians via efficient attribute transformation,” in ECCV, 2024.
- [19] H. Huang, W. Huang, Q. Yang, Y. Xu, and Z. Li, “A hierarchical compression technique for 3D Gaussian splatting compression,” in ICASSP, 2025.
- [20] F. Di Sario, R. Renzulli, M. Grangetto, A. Sugimoto, and E. Tartaglione, “GoDe: Gaussians on demand for progressive level of detail and scalable compression,” arXiv:2501.13558, 2025.
- [21] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, “Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering,” in CVPR, 2024.
- [22] Y. Wang, Z. Li, L. Guo, W. Yang, A. Kot, and B. Wen, “ContextGS: Compact 3D Gaussian splatting with anchor level context model,” in NeurIPS, 2024.
- [23] Y. T. Zhan et al., “CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression,” in ICLR, 2025.
- [24] Y. Chen et al., “PCGS: Progressive compression of 3D Gaussian splatting,” in AAAI, 2025.
- [25] T. Wang, Z. Yu, and Y. Xu, “TC-GS: Tri-plane based compression for 3D Gaussian splatting,” in ICME, 2025.
- [26] H. Xu, X. Wu, and X. Zhang, “3DGS compression with sparsity-guided hierarchical transform coding,” arXiv:2505.22908, 2025.
- [27] Z. Tang et al., “NeuralGS: Bridging neural fields and 3D Gaussian splatting for compact 3D representations,” in AAAI, 2025.
- [28] L. Liu, Z. Chen, and D. Xu, “3D Gaussian splatting data compression with mixture of priors,” in ACM MM, 2025.

참 고 문 헌

-
- [29] E. Ververas, R. A. Potamias, J. Song, J. Deng, and S. Zafeiriou, “SAGS: Structure-aware 3D Gaussian splatting,” in ECCV, 2024.
- [30] G. Fang and B. Wang, “Mini-Splatting: Representing scenes with a constrained number of Gaussians,” in ECCV, 2024.
- [31] H. Li, J. Liu, M. Szaier, and O. Camps, “3D-HGS: 3D half-Gaussian splatting,” in CVPR, 2025.
- [32] S. Javed, A. J. Khan, C. Dumery, C. Zhao, and M. Salzmann, “Temporally compressed 3D Gaussian splatting for dynamic scenes,” in BMVC, 2025.
- [33] A. Tu, H. Ying, A. Hanson, Y. Lee, T. Goldstein, and M. Zwicker, “Speedy deformable 3D Gaussian splatting: Fast rendering and compression of dynamic scenes,” arXiv:2506.07917, 2025.
- [34] Y. Yuan, Q. Shen, X. Yang, and X. Wang, “1000+ FPS 4D Gaussian splatting for dynamic scene rendering,” in NeurIPS, 2025.
- [35] J. Lee, C.-Y. Won, H. Jung, I. Bae, and H.-G. Jeon, “Fully explicit dynamic Gaussian splatting,” in NeurIPS, 2024.
- [36] J. Sun, H. Jiao, G. Li, Z. Zhang, L. Zhao, and W. Xing, “3DGStream: On-the-fly training of 3D Gaussians for efficient streaming of photo-realistic free-viewpoint videos,” in CVPR, 2024.
- [37] Z. Luo, H. Ran, and L. Lu, “Instant4D: 4D Gaussian splatting in minutes,” in NeurIPS, 2025.
- [38] S. Oh, Y. Lee, H. Jeon, and E. Park, “Hybrid 3D-4D Gaussian splatting for fast dynamic scene representation,” arXiv:2505.13215, 2025.
- [39] Y. Jiang et al., “HiFi4G: High-fidelity human performance rendering via compact Gaussian splatting,” in CVPR, 2024.
- [40] X. Zhang et al., “MEGA: Memory-efficient 4D Gaussian splatting for dynamic scenes,” in ICCV, 2025.
- [41] J. Lei, Y. Weng, A. Harley, L. Guibas, and K. Daniilidis, “MoSca: Dynamic Gaussian fusion from casual videos via 4D motion scaffolds,” in CVPR, 2025.
- [42] H. Kong, X. Yang, and X. Wang, “Efficient Gaussian splatting for monocular dynamic scene rendering via sparse time-variant attribute modeling,” in AAAI, 2025.
- [43] S. Kwak, J. Kim, J. Y. Jeong, W.-S. Cheong, J. Oh, and M. Kim, “MoDec-GS: Global-to-local motion decomposition and temporal interval adjustment for compact dynamic 3D Gaussian splatting,” in CVPR, 2025.
- [44] J. Chen et al., “HAIF-GS: Hierarchical and induced flow-guided Gaussian splatting for dynamic scene,” in NeurIPS, 2025.
- [45] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin, “Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction,” in CVPR, 2024.
- [46] G. Wu et al., “4D Gaussian splatting for real-time dynamic scene rendering,” in CVPR, 2024.
- [47] B. P. Duisterhof et al., “DeformGS: Scene flow in highly deformable scenes for deformable object manipulation,” in WAFR, 2024.
- [48] Z. Li, Z. Chen, Z. Li, and Y. Xu, “Spacetime Gaussian feature splatting for real-time dynamic view synthesis,” in CVPR, 2024.
- [49] J. Yang, W. Su, S. Zhang, Y. Han, J. Suo, and Q. Zhang, “Scale-GS: Efficient scalable Gaussian splatting via redundancy-filtering training on streaming content,” arXiv:2508.21444, 2025.
- [50] Z. Zheng et al., “4DGCPro: Efficient hierarchical 4D Gaussian compression for progressive volumetric video streaming,” in NeurIPS, 2025.
- [51] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, “Tanks and Temples: Benchmarking large-scale scene reconstruction,” ACM ToG, 2017.
- [52] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, “Deep blending for free-viewpoint image-based rendering,” ACM ToG, 2018.
- [53] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, “BungeeNeRF: Progressive neural radiance field for extreme multi-scale scene rendering,” in ECCV, 2022.
- [54] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Mip-NeRF 360: Unbounded anti-aliased neural radiance fields,” in CVPR, 2022.
- [55] N. Sabater et al., “Dataset and pipeline for multi-view light-field video,” in CVPRW, 2017.
- [56] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, “D-NeRF: Neural radiance fields for dynamic scenes,” in CVPR, 2021.
- [57] K. Park et al., “HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields,” ACM ToG, 2021.
- [58] T. Li et al., “Neural 3D video synthesis from multi-view video,” in CVPR, 2022.
- [59] Z. Yan, C. Li, and G. H. Lee, “NeRF-DS: Neural radiance fields for dynamic specular objects,” in CVPR, 2023.
- [60] 피규어 출처: S. Youn, S. Lee, G. Kim, W. Kwon, S.-H. Bae, and J. Oh, “SUCCESS-GS: Survey of Compactness and Compression for Efficient Static and Dynamic Gaussian Splatting,” arXiv, 2025.
-

저 자 소 개



윤 석 현

- 2025년 : 한경국립대학교 전기전자제어공학부 학사
- 2025년 ~ 현재 : 중앙대학교 첨단영상대학원 메타버스융합학과 석사과정
- 주관심분야 : 3D Vision, Low-Level Vision



오 지 형

- 2017년 : KAIST 전기공학부 학사
- 2019년 : KAIST 전기공학부 석사
- 2023년 : KAIST 전기공학부 박사
- 2023년 ~ 현재 : 중앙대학교 첨단영상대학원 조교수
- 2024년 ~ 현재 : 인쇼츠 (주) CAIO
- 주관심분야 : 3D Vision, Low-Level Vision, Generative AI