

SUCCESS-GS: Survey of Compactness and Compression for Efficient Static and Dynamic Gaussian Splatting

SEOKHYUN YOUN*, Chung-Ang University, South Korea

SOOHYUN LEE*, Kyung Hee University, South Korea

GEONHO KIM*, Chung-Ang University, South Korea

WEEYOUNG KWON, Chung-Ang University, South Korea

SUNG-HO BAE[†], Kyung Hee University, South Korea

JIHYONG OH[†], Chung-Ang University, South Korea

<https://cmlab-korea.github.io/Awesome-Efficient-GS/>

3D Gaussian Splatting (3DGS) has emerged as a powerful explicit representation enabling real-time, high-fidelity 3D reconstruction and novel view synthesis. However, its practical use is hindered by the massive memory and computational demands required to store and render millions of Gaussians. These challenges become even more severe in 4D dynamic scenes. To address these issues, the field of Efficient Gaussian Splatting has rapidly evolved, proposing methods that reduce redundancy while preserving reconstruction quality. This survey provides the first unified overview of efficient 3D and 4D Gaussian Splatting techniques. For both 3D and 4D settings, we systematically categorize existing methods into two major directions, Parameter Compression and Restructuring Compression, and comprehensively summarize the core ideas and methodological trends within each category. We further cover widely used datasets, evaluation metrics, and representative benchmark comparisons. Finally, we discuss current limitations and outline promising research directions toward scalable, compact, and real-time Gaussian Splatting for both static and dynamic 3D scene representation.

1 INTRODUCTION

3D scene representation and photorealistic novel view synthesis are crucial for a wide range of applications, including virtual reality (VR) [17, 115], augmented reality (AR) [96], media generation [75], autonomous driving [110], and large-scale visualization [92, 98]. The evolution of these tasks has been dramatically accelerated by the emergence of Neural Radiance Fields (NeRF) [6]. By directly training *implicit* neural networks through differentiable volume rendering, NeRF has achieved high-quality 3D reconstruction from 2D images. However, despite its impressive visual quality, NeRF suffers from extremely slow training rendering speed, since it requires a large number of ray samples per pixel to generate a single image. [18, 37, 55, 70, 80, 89, 128]

To overcome the limitations of such implicit representations, 3D Gaussian Splatting (3DGS) [42] revisits the idea of *explicit* 3D scene representation. Similar to traditional 3D representations such as point clouds [77, 78, 82] and meshes [24, 122], 3DGS explicitly represents a scene using a collection of primitives. Unlike these classical forms, however, 3DGS models the scene as a set of anisotropic 3D Gaussians, each parameterized by a set of learnable attributes including position, scale, rotation, opacity, and view-dependent color coefficients (see Sec. 2.1 for more details). By

*These authors contributed equally to this paper.

[†]Co-corresponding authors.

Authors' Contact Information: Seokhyun Youn, hisn16@cau.ac.kr, Chung-Ang University, South Korea; Soohyun Lee, nata1225@khu.ac.kr, Kyung Hee University, South Korea; Geonho Kim, joelkimgh@cau.ac.kr, Chung-Ang University, South Korea; Weeyoung Kwon, weeyoungkwon@cau.ac.kr, Chung-Ang University, South Korea; Sung-Ho Bae, Kyung Hee University, Seoul, South Korea, shbae@khu.ac.kr; Jihyong Oh, Chung-Ang University, Seoul, South Korea, jihyongoh@cau.ac.kr.

learning these attributes directly from multi-view images, 3DGS not only achieves high-quality scene reconstruction, but also enables real-time rendering performance, which is infeasible for NeRF-based approaches.

Further, recent methods can reconstruct a 4D scene representation from monocular or multi-view video inputs. Dynamic 3D/4D Gaussian Splatting extends conventional static 3DGS into the temporal dimension, enabling the reconstruction of not only static scenes and objects but also dynamic ones exhibiting non-rigid motion. By modeling spatial and temporal variations within a unified spatio-temporal Gaussian framework, 4DGS provides continuous motion representation and temporally coherent rendering (see Sec. 2.2 for more details). This capability opens up new possibilities for dynamic scene editing [46], motion capture [41], and realistic simulation in robotics and autonomous driving [65], where time-varying geometry and appearance are crucial.

However, despite these advantages, the practical deployment of Gaussian Splatting-based methods remains challenging due to their massive memory footprint and computational overhead [73]. A typical high-resolution static scene often contains millions of 3D Gaussians, which require significantly more memory than NeRF-based models [22]. Furthermore, 4D scene reconstruction with Gaussian Splatting demands even greater memory consumption, as each Gaussian must encode temporal information across multiple frames [125].

To overcome these bottlenecks, a new research direction has emerged under the umbrella of Efficient Gaussian Splatting, aiming to reduce the redundancy of Gaussian primitives while preserving rendering quality. As various studies have been conducted, research on Efficient Gaussian Splatting can be categorized into the following groups:

- **Parameter Compression (Sec. 3.1, Sec. 4.1):** This approach directly compresses the attributes of Gaussians using techniques such as pruning, quantization, and entropy coding.
- **Restructuring Compression (Sec. 3.2, Sec. 4.2):** This approach performs compression by redesigning the structure of the original Gaussian Splatting framework through methods such as hierarchical anchors, neural integration, or geometry-aware organization.

This survey provides the first comprehensive overview of Efficient 3D and 4D Gaussian Splatting (Sec. 3, Sec. 4). For both static 3DGS and dynamic 3D/4D Gaussian Splatting, we categorize existing methods into Parameter Compression (Sec. 3.1, Sec. 4.1) and Restructuring Compression (Sec. 3.2, Sec. 4.2), further organizing them into subcategories. For each class, we summarize the core methodology, mathematical formulation, and representative works. In addition, this survey provides a systematic overview of widely used datasets and evaluation metrics in Gaussian Splatting research, and presents comparisons of recent methods to establish a fair and consistent basis for performance evaluation (Sec. 5). Finally, we comprehensively discuss the limitations of existing approaches and propose key future directions that may guide the development of more efficient Gaussian Splatting techniques (Sec. 6). By establishing a unified taxonomy and connecting isolated research threads, this survey aims to serve as a foundation for the further development of scalable, efficient, and robust Gaussian Splatting frameworks for static and dynamic 3D scene representation.

2 PRELIMINARY

All mathematical notations used in this survey are summarized in the supplementary material.

2.1 Static 3D Gaussian Splatting (3DGS)

3DGS represents a 3D scene as a set of anisotropic 3D Gaussians and directly projects them into 2D screen space for rendering. Unlike Implicit Neural Representations (e.g., NeRF [6]), 3DGS can be optimized without neural networks and

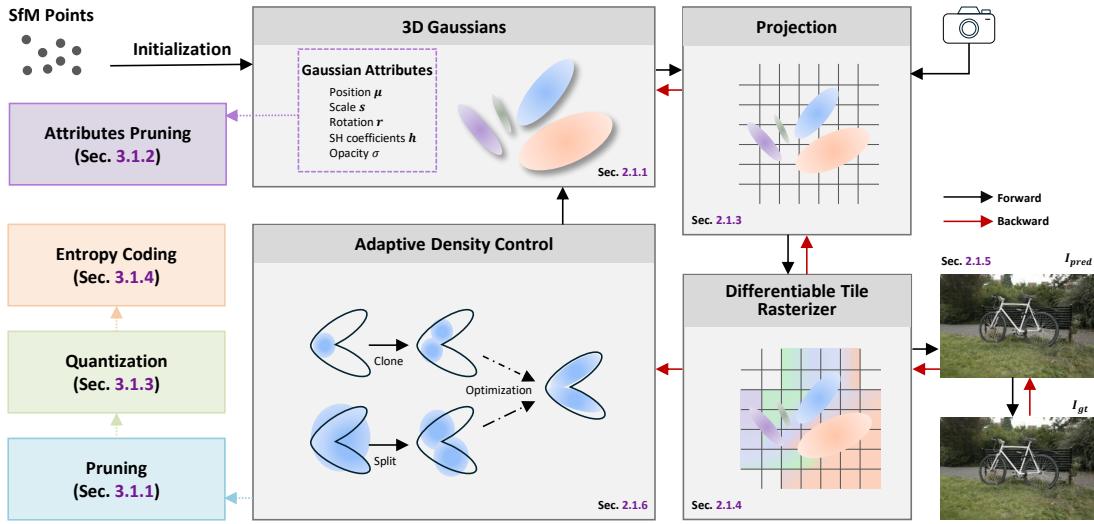


Fig. 1. Overview of the static 3D Gaussian Splatting (3DGS) pipeline. A scene is represented as a set of 3D Gaussians with attributes including position μ , scale s , rotation r , spherical harmonics (SH) coefficients h , and opacity σ (Sec. 2.1.1). Each Gaussian is projected into the camera coordinate system and rendered onto the image plane through a differentiable tile rasterizer (Sec. 2.1.3, Sec. 2.1.4). The rendering is optimized by minimizing the difference between predicted and GT images (Sec. 2.1.5). To improve reconstruction quality, adaptive density control clones or splits Gaussians based on view-space gradient signals while pruning nearly transparent Gaussians (Sec. 2.1.6). In addition, efficiency-oriented strategies such as pruning, quantization, and entropy coding are applied to compress Gaussian attributes (Sec. 3.1).

supports real-time rendering through a CUDA-based tile rasterization pipeline. This section describes the initialization, rendering pipeline, and optimization process of 3DGS. An overall overview of the 3DGS framework is shown in Fig. 1.

2.1.1 Initialization and Attribute Definition. 3DGS requires multi-view RGB images of the same scene along with their corresponding camera pose information as input. Therefore, before the actual initialization of the 3D Gaussians, a Structure-from-Motion (SfM) tool (e.g., COLMAP [85]) is used to initialize the sparse point cloud of the scene and recover the camera poses for each image. The sparse point cloud is then used for Gaussian initialization, where each point is initialized as a single 3D Gaussian. The estimated camera poses are used later in the projection stage.

Each Gaussian has the following attributes:

- **Position $\mu \in \mathbb{R}^3$:** It represents the center coordinates of the 3D Gaussian. This attribute is initialized using the world coordinates from the sparse point cloud.
- **Scale $s \in \mathbb{R}^3$:** It indicates the spatial extent occupied by the 3D Gaussian.
- **Rotation $r \in \mathbb{R}^4$:** It represents the orientation of the 3D Gaussian, initialized as a unit quaternion.
- **View-dependent Spherical Harmonics (SH) coefficients $h \in \mathbb{R}^{(d+1)^2 \times 3}$:** They represent colors that vary with the viewing direction. d is a parameter that denotes the degree of the SH, which determines the level of detail in representing view-dependent colors in the scene. It is commonly set to $d = 3$ in practice. Since the spherical harmonics functions correspond to each RGB channel, the SH coefficients are also defined accordingly.

- **Opacity** $\sigma \in \mathbb{R}$: Each 3D Gaussian has an opacity value in the range $[0, 1]$, stored with M -bit precision. As the opacity value approaches 0, the 3D Gaussian becomes more transparent, while as it approaches 1, it becomes more opaque. In the original 3DGS paper [42], it uses $M = 8$ to store Gaussian opacity.

Based on the above, in this survey, the set of N 3D Gaussians \mathcal{G} at a specific camera pose and their corresponding attributes are denoted as follows:

$$\mathcal{G} = \{G_i = (\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{h}_i, \sigma_i)\}_{i=1}^N. \quad (1)$$

Let G_i denote the i -th Gaussian among N Gaussians, with position $(\boldsymbol{\mu}_i)$, scale (\mathbf{s}_i) , rotation (\mathbf{r}_i) , SH coefficients (\mathbf{h}_i) , and opacity (σ_i) . Each attribute of the Gaussian is optimized through the training process.

2.1.2 Mathematical Definition of a 3D Gaussian. When the center vector of the i -th Gaussian in coordinate system is $\boldsymbol{\mu}_i$, the Gaussian is defined as

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

where \mathbf{x} is an arbitrary point in the world coordinate and Σ_i denotes the covariance matrix of the i -th Gaussian. The covariance matrix is formulated as

$$\Sigma_i = \mathbf{R}_i \mathbf{S}_i \mathbf{S}_i^\top \mathbf{R}_i^\top, \quad (3)$$

where \mathbf{R}_i and \mathbf{S}_i are the rotation and scaling matrices of the Gaussian, respectively, which are derived from the Gaussian's attributes \mathbf{r}_i and \mathbf{s}_i .

2.1.3 Projection. Let \mathbf{W} be the coordinate transformation from the world coordinates to the camera coordinates, and \mathbf{J} be the Jacobian of the camera coordinate to screen coordinate projection. The covariance matrix is formulated as:

$$\Sigma'_i = \mathbf{J} \mathbf{W} \Sigma_i \mathbf{W}^\top \mathbf{J}^\top. \quad (4)$$

This results in the representation of each 3D anisotropic Gaussian as a 2D ellipse in the screen coordinate system.

2.1.4 Differentiable Tile Rasterization. The rendering pipeline proceeds as follows. First, the image is divided into $T \times T$ pixel tiles. According to the original 3DGS paper, the tile size T was set to 16. Next, for each projected Gaussian, the overlapping tiles are determined. Finally, per-pixel rasterization is performed to compute the contribution of each Gaussian to the pixels within the overlapping tiles.

For each Gaussian G_i , the alpha value at pixel \mathbf{p} is given by

$$\alpha_i(\mathbf{p}) = \sigma_i \cdot g_i(\mathbf{p}), \quad (5)$$

where $g_i(\mathbf{p})$ denotes the value of the projected 2D Gaussian at pixel \mathbf{p} , computed as

$$g_i(\mathbf{p}) = \exp\left(-\frac{1}{2}(\mathbf{p} - \boldsymbol{\mu}'_i)^\top (\Sigma'_i)^{-1}(\mathbf{p} - \boldsymbol{\mu}'_i)\right). \quad (6)$$

Here, $\boldsymbol{\mu}'_i$ denotes the 2D projected center of the i -th Gaussian after applying the camera projection, and Σ'_i is the 2D covariance matrix obtained by projecting the 3D covariance Σ_i using the Jacobian of the projection function.

The final color of pixel \mathbf{p} is obtained via alpha blending, which combines the color \mathbf{c}_i predicted from the Gaussian's SH coefficients with the corresponding depth information:

$$C(\mathbf{p}) = \sum_{i \in N} \mathbf{c}_i \alpha_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{p})). \quad (7)$$

In this expression, N denotes the total number of Gaussians projected onto the pixel. c_i is the view-dependent base color of the Gaussian, computed from its SH coefficients and the camera viewing direction. The rasterizer is fully differentiable, allowing gradients to be backpropagated to all Gaussian parameters.

2.1.5 Optimization. The reconstruction is optimized by minimizing the difference between the predicted 2D image I_{pred} and the ground truth (GT) I_{gt} :

$$L = (1 - \lambda) \cdot L_1(I_{pred}, I_{gt}) + \lambda \cdot (1 - SSIM(I_{pred}, I_{gt})), \quad (8)$$

where $L_1(\cdot)$ is the pixel-wise mean absolute error, and $SSIM$ [108] is the structural similarity index. λ is a weighting factor that controls the relative contribution of the $L_1(\cdot)$ and the $SSIM$ loss in the total loss function.

2.1.6 Adaptive Density Control. Since the initial Gaussians are generated from a sparse point cloud obtained via SfM, they are insufficient to represent the entire scene. Therefore, during training, the number of Gaussians is adaptively increased by duplicating or splitting existing Gaussians. To select Gaussians for densification, the view-space position gradients of the Gaussians are considered. A large gradient indicates a high reconstruction error in the corresponding region, implying that more Gaussians are required to represent the scene in greater detail. Specifically, if the view-space position gradients of a Gaussian exceed a certain threshold τ_{pos} , which is set to 0.002 in the 3DGS paper, densification is performed. In this process, Gaussians with relatively small scales s are cloned, while Gaussians with larger scales are split into smaller Gaussians and distributed around the original position. This densification step is performed at fixed epoch intervals, which reduces unnecessary computational cost and ensures efficient training. In addition, if the opacity σ_i of a Gaussian is less than a certain threshold ϵ_σ , it is considered nearly transparent and thus negligible for scene reconstruction, and such Gaussians are pruned. This prevents an uncontrolled growth in the number of Gaussians.

2.2 Dynamic 3D/4D Gaussian Splatting

In recent years, research interest has shifted beyond static scene reconstruction using 3D Gaussian Splatting toward the reconstruction of dynamic scenes that incorporate a temporal dimension. As this field has gained momentum, various approaches have been proposed to extend 3D Gaussians for modeling dynamic scenes. In this section, we classify and describe how different dynamic 3DGS and 4DGS studies represent dynamic scenes. Representative approaches for dynamic 3D/4D Gaussian Splatting are illustrated in Fig. 2. It should be noted that many works use the terms **dynamic 3DGS** [63] and **4DGS** [109, 119] interchangeably, and there is ongoing debate regarding which term to adopt. In this survey, we do not distinguish between the two terms.

Some models adopt deformation modeling with a canonical 3D Gaussian set. **Deformable 3DGS** [97] initializes all 3D Gaussians in a canonical state and models their dynamic motion with a deformation network. The deformation network is a time-conditioned MLP that produces offsets for position, rotation, and scale:

$$(\Delta\mathbf{x}, \Delta\mathbf{r}, \Delta\mathbf{s}) = F_\theta(\mathbf{x}_c, t), \quad (9)$$

where \mathbf{x}_c is the canonical position and t is the timestamp.

This design provides flexibility in modeling non-rigid motion and allows monocular dynamic scene reconstruction. However, since it relies on a dense MLP, the approach can suffer from temporal jitter, especially when supervision is limited. To mitigate this issue, Deformable 3DGS introduces Annealing Smooth Training (AST), which gradually regularizes the deformation field during optimization and effectively stabilizes the temporal dynamics.

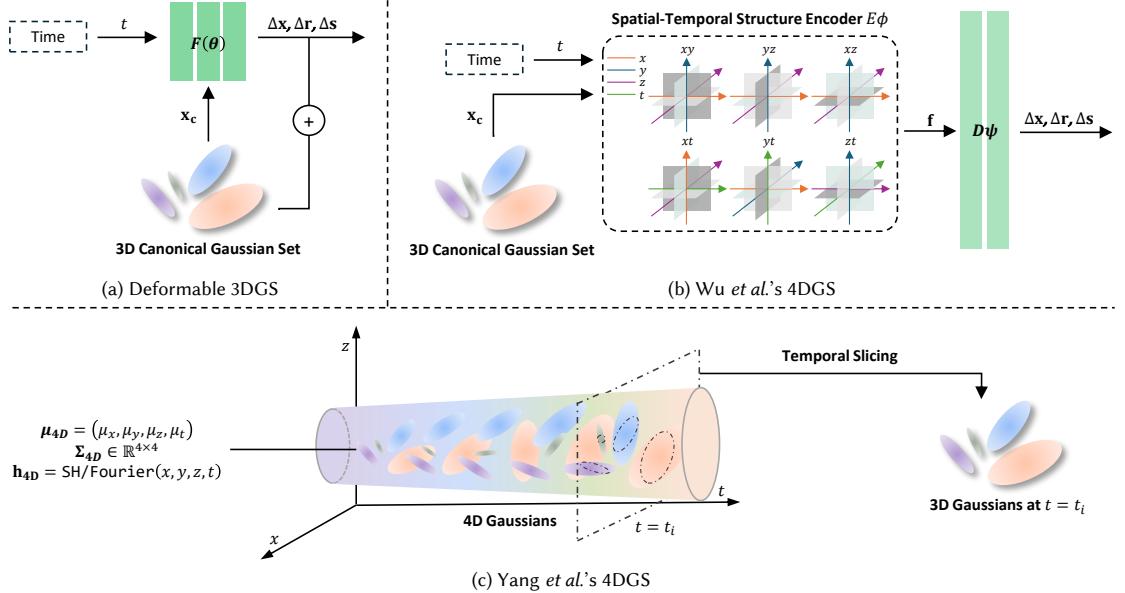


Fig. 2. Overview of representative approaches for dynamic 3D/4D Gaussian Splatting. (a) **Deformable 3DGs** [97] initializes a canonical set of 3D Gaussians and models temporal motion using a time-conditioned MLP that predicts offsets in position, rotation, and scale. (b) **Wu et al.’s 4DGs** [109] adopts a structured encoder-decoder design, where a spatial-temporal structure encoder (HexPlane) organizes the 4D domain (x, y, z, t) into multi-plane features, which are then decoded into Gaussian attribute offsets. (c) **Yang et al.’s 4DGs** [119] directly extends 3D Gaussians into the spatio-temporal domain by defining each Gaussian as a 4D primitive with a 4D mean, covariance, and appearance coefficients. During rendering, 4D Gaussians are converted into 3D Gaussians at target timestamps through temporal slicing.

Wu et al.’s 4DGS [109] extends the canonical Gaussian formulation with a structured encoder-decoder design. Instead of directly predicting offsets with a simple MLP, a spatial-temporal structure encoder organizes the 4D domain (x, y, z, t) into multi-plane features. These features are interpolated to produce a latent code:

$$\mathbf{f} = E_\phi(\mathbf{x}_c, t), \quad (10)$$

where E_ϕ denotes the HexPlane [9] encoder. The latent feature f is then decoded into Gaussian attribute offsets:

$$(\Delta x, \Delta r, \Delta s) = D_\psi(f), \quad (11)$$

where D_ψ is a lightweight MLP decoder network. This grid-like representation improves expressiveness and motion fidelity compared to MLP-only approaches, but it requires more memory and results in slower rendering speed due to the cost of querying large spatio-temporal feature grids.

Yang et al.'s 4DGs [119] directly extends 3DGS into the spatio-temporal domain by defining each Gaussian as a four-dimensional primitive in (x, y, z, t) . In this formulation, time t is not treated as a conditional input but rather as an independent coordinate dimension. Each Gaussian is parameterized with a 4D mean, a 4D covariance, and 4D appearance coefficients:

$$\mu_{4D} = (\mu_x, \mu_y, \mu_z, \mu_t), \quad \Sigma_{4D} \in \mathbb{R}^{4 \times 4}, \quad \mathbf{h}_{4D} = \text{SH/Fourier}(x, y, z, t). \quad (12)$$

Here, μ_{4D} denotes the 4D position, Σ_{4D} is the 4D covariance matrix (encoding both scale and rotation), and \mathbf{h}_{4D} represents spherical harmonics or Fourier coefficients defined over the 4D coordinates. During rendering, 4D Gaussians must be converted into valid 3D Gaussians corresponding to the target timestamp t_i . This is achieved through temporal slicing, where each 4D Gaussian is projected onto the hyperplane $t = t_i$. Unlike deformation network approaches, this method represents Gaussians as intrinsic 4D primitives that evolve continuously in time, allowing it to capture complex and periodic dynamics. For photorealistic rendering, Fourier basis functions are incorporated into the appearance coefficients to model high-frequency temporal variations.

3 STATIC

Static 3DGS requires a significant number of 3D Gaussians \mathcal{G} to maintain high fidelity. This leads to large memory and storage requirements (often over 1GB for large real-world scenes [49]). This makes deployment difficult in resource-constrained environments like portable devices or head-mounted displays [2]. Therefore, reducing storage space and memory usage is essential to improve the scalability of 3DGS applications [69]. To this end, efficient Static 3DGS approaches can be categorized into two main directions:

- **Parameter Compression:** These approaches compress the 3D Gaussians without modifying the original 3DGS [42] model architecture.
- **Restructuring Compression:** These approaches fundamentally modify the original 3DGS [42] model architecture to obtain an efficient scene representation.

3.1 Parameter Compression

Parameter Compression aims to reduce storage space and memory usage without modifying the 3DGS [42] model architecture. It can be applied to trained 3DGS models, making it flexible in various scenarios. This survey classifies Parameter Compression methods into five main strategies:

- **Pruning:** This strategy removes redundant or low-contribution 3D Gaussians based on various criteria.
- **Attribute Pruning:** This strategy simplifies Gaussian attributes by compressing specific components.
- **Quantization:** This strategy discretizes Gaussian attributes by reducing their bit precision.
- **Entropy Coding:** This strategy utilizes statistical redundancy in quantized attributes to minimize storage.
- **Structured Compression:** This strategy organizes 3D Gaussians by spatial relationships to improve efficiency.

3.1.1 Pruning. **Pruning** is the most fundamental approach to reducing redundant 3D Gaussians. To minimize the redundant 3D Gaussians, the original 3DGS [42] periodically resets the opacity of 3D Gaussians to a low value at specific intervals and prunes Gaussians that remain below an opacity threshold after a certain number of iterations (Sec. 2.1.6). However, according to Lee *et al.* [49], relying solely on this opacity-based control method still results in a large number of redundant 3D Gaussians. To this end, the pruning methods use various criteria to eliminate these redundancies.

Learnable mask-based Pruning uses learnable parameters to automatically select which 3D Gaussians to remove. Lee *et al.* [49] use this approach to enable end-to-end learning of pruning decisions. It considers both scale s and opacity σ together, avoiding the fixed thresholds used in the original 3DGS [42]. The binary mask $\mathcal{M}_n \in \{0, 1\}$ for the n -th 3D Gaussian is computed as follows through the learnable mask parameter $m_n \in [0, 1]$:

$$\mathcal{M}_n = \text{sg}(\mathbb{1}[\text{Sig}(m_n) > \epsilon] - \text{Sig}(m_n)) + \text{Sig}(m_n), \quad (13)$$

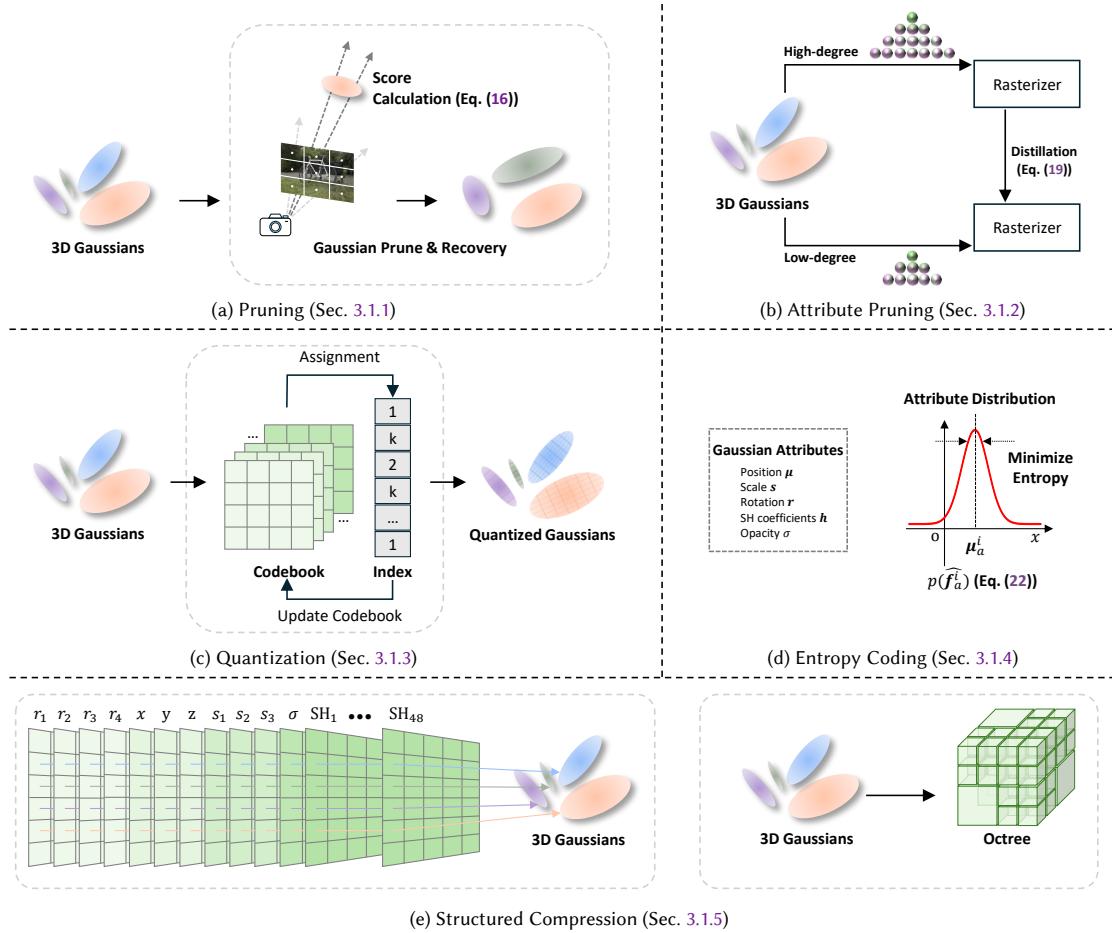


Fig. 3. Overview of Parameter Compression strategies for Static 3DGS. These approaches reduce redundancy in the 3DGS representations without modifying the 3DGS [42] model architecture. (a) **Pruning** (Sec. 3.1.1) removes redundant 3D Gaussians. (b) **Attribute Pruning** (Sec. 3.1.2) compresses specific Gaussian attributes. (c) **Quantization** (Sec. 3.1.3) reduces Gaussian attribute bit precision. (d) **Entropy Coding** (Sec. 3.1.4) compresses quantized Gaussian attributes by exploiting statistical redundancy. (e) **Structured Compression** (Sec. 3.1.5) organizes 3D Gaussians by spatial relationships to improve compression efficiency.

where $\text{sg}(\cdot)$ denotes the Stop Gradient operator, $\mathbb{1}[\cdot]$ denotes the indicator function, and $\text{Sig}(\cdot)$ denotes the Sigmoid activation function. The scale \hat{s} and opacity $\hat{\sigma}$ of the n -th masked 3D Gaussian are calculated as follows:

$$\hat{s}_n = \mathcal{M}_n s_n, \quad \hat{\sigma}_n = \mathcal{M}_n \sigma_n. \quad (14)$$

Additionally, Lee *et al.* regularize the model to minimize the redundant 3D Gaussians G through masking loss L_m :

$$L_m = \frac{1}{N} \sum_{n=1}^N \text{Sig}(m_n). \quad (15)$$

Significance Score-based Pruning methods introduce scoring functions to quantify the importance of each 3D Gaussian for scene reconstruction. LightGaussian [22] proposes a Global Significance (GS) score that comprehensively

evaluates the contribution of each 3D Gaussian [26], inspired by the rendering equation Eq. (7). As shown in Fig. 3, the GS score for the j -th 3D Gaussian is calculated as follows:

$$GS_j = \sum_{i=1}^{|I| \times P_i} \mathbb{1}[G_j \wedge r_i \neq \emptyset] \cdot \sigma_j \cdot \prod_{k=1}^{j-1} (1 - \sigma_k) \cdot \gamma(\Sigma_j), \quad (16)$$

where $|I|$ denotes the total number of images in the training set, and P_i denotes the number of pixels in the i -th image. The $\mathbb{1}[G_j \wedge r_i \neq \emptyset]$ term represents whether the ray from the i -th pixel r_i intersects the j -th 3D Gaussian G_j . The $\gamma(\Sigma_j)$ term represents the normalized volume of the j -th 3D Gaussian. EAGLES [26] introduces a new criterion to identify inefficient 3D Gaussians. It calculates the influence $W_{i,p}$ of the i -th 3D Gaussian at a specific pixel p , and the total influence W_i as follows:

$$W_{i,p} = \alpha_i T_i = \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad W_i = \sum_p W_{i,p}. \quad (17)$$

MesonGS [113] calculates the significance score for each 3D Gaussian by multiplying two components: a view-dependent significance score W_i from Eq. (17) and a volume-based view-independent significance score [38, 112]. OMG [48] extends Eq. (17) by additionally considering the following Local Distinctiveness:

$$\text{Local Distinctiveness} = \left(\frac{1}{K} \sum_{j \in \mathcal{N}_i^K} \|T_i - T_j\|_1 \right)^\lambda, \quad (18)$$

where \mathcal{N}_i^K denotes the set of the k -Nearest Neighbors (KNN) of the i -th 3D Gaussian and $\frac{1}{K} \sum_{j \in \mathcal{N}_i^K} \|T_i - T_j\|_1$ represents the average difference in the appearance features T .

Gradient-based Pruning methods calculate the importance of 3D Gaussians based on their gradient magnitudes. PUP 3DGS [33] measures 3D Gaussian significance using a sensitivity score derived from the diagonal components of the Hessian. Speedy-Splat [32] reparameterizes the Hessian approximation of PUP 3DGS to reduce storage requirements. Trimming the Fat [3] and ELMGS [2] propose Gradient Aware Pruning (GAP), which uses both opacity and gradient to prune inefficient 3D Gaussians.

3.1.2 Attribute Pruning. **Attribute pruning** simplifies Gaussian attributes by selectively removing or compressing components that have a minimal impact on visual fidelity. LightGaussian [22] uses knowledge distillation to reduce the dimension of SH coefficients \mathbf{h} while maintaining visual fidelity. As shown in Fig. 3, the student model is trained to match the output of the teacher model under the same camera pose $[R|t]$.

$$\mathcal{L}_{\text{distill}} = \frac{1}{P_i} \sum_{i=1}^{P_i} \|C_{\text{teacher}}(r_i; [R|t]) - C_{\text{student}}(r_i; [R|t])\|^2. \quad (19)$$

RDO-Gaussian [103] applies the masking strategy from Eq. (13) to the certain Gaussian attributes.

3.1.3 Quantization. **Quantization** discretizes Gaussian attributes by reducing their bit precision while maintaining visual fidelity [112]. Niedermayr *et al.* [71] define sensitivity $S(p)$ to quantify the impact of scalar attribute p on total energy E_j across the training images:

$$S(p) = \frac{1}{\sum_{i=1}^{|I|} P_i} \sum_{j=1}^{|I|} \left| \frac{\partial E_j}{\partial p} \right|. \quad (20)$$

Based on Eq. (20), they introduce sensitivity-aware k -means clustering to the covariance matrix Σ and SH coefficients \mathbf{h} . Lee *et al.* [49] use Residual Vector Quantization (R-VQ) to quantize Gaussian attributes. LightGaussian [22, 69] applies Vector Quantization (VQ) to the SH coefficients \mathbf{h} with low GS scores, as defined in Eq. (16). CompGS [69] compresses Gaussian attributes using VQ. RDO-Gaussian [103] modifies the codeword selection process in VQ to jointly minimize rate loss $r_{i,k}$ and distortion loss $d_{i,k}$ for codeword index k . For example, the codeword selection for the scale parameter s_i of the i -th 3D Gaussian is defined as follows:

$$j = \arg \min_k \frac{r_{i,k}^{(s)}}{\lambda^{(s)}} + d_{i,k}^{(s)} = \arg \min_k -\frac{\log p_k}{\lambda^{(s)}} + (s_i - \mathbf{CB}^{(s)}[k])^2. \quad (21)$$

SizeGS [112] introduces a size estimator which establishes a relationship between hyperparameters and compressed file size. This relationship guides a Hierarchical Mixed Precision Quantization (H-MPQ) for Gaussian attributes, which applies different quantization precision levels to each attribute. FlexGaussian [95] proposes a training-free compression method that combines mixed-precision quantization and attribute-discriminative pruning. This approach compresses 3DGS models without retraining.

3.1.4 Entropy Coding. **Entropy coding** utilizes statistical redundancy in quantized Gaussian attributes to minimize storage requirements. This strategy adaptively compresses a scene representation by utilizing statistical patterns. Subsequent methods are particularly effective for 3DGS compression as quantized Gaussian attributes often exhibit high statistical redundancy. Niedermayr *et al.* [71] use the DEFLATE algorithm [19] to compress codebook indices. RDO-Gaussian [103] compresses the indices obtained from Eq. (21) using arithmetic coding. HAC [14] models quantized anchor attributes \hat{f}_a as a Gaussian distribution ϕ_{μ_a, σ_a} based on empirical observations. Specifically, as shown in Fig. 3, the probability of the quantized anchor features for the i -th anchor $p(\hat{f}_a^i)$ is calculated as follows:

$$\begin{aligned} p(\hat{f}_a^i) &= \int_{\hat{f}_a^i - \frac{1}{2}q_i}^{\hat{f}_a^i + \frac{1}{2}q_i} \phi_{\mu_a^i, \sigma_a^i}(x) dx \\ &= \Phi_{\mu_a^i, \sigma_a^i}\left(\hat{f}_a^i + \frac{1}{2}q_i\right) - \Phi_{\mu_a^i, \sigma_a^i}\left(\hat{f}_a^i - \frac{1}{2}q_i\right), \\ \mu_a^i, \sigma_a^i &= \text{MLP}_c(f_h^i), \end{aligned} \quad (22)$$

where q_i denotes the adaptive quantization step size for the attribute of the i -th anchor, f_h^i represents the interpolated hash features, and $\text{MLP}_c(f_h^i)$ is a Multi-layer Perceptron (MLP) that utilizes the interpolated hash features f_h^i to predict the mean μ_a^i and standard deviation σ_a^i of the distribution of the quantized anchor attributes. FCGS [13] compresses 3DGS using a single forward pass, avoiding the scene-specific optimization process. This framework utilizes a Multi-path Entropy Module (MEM) to compress Gaussian attributes, trading off size and visual fidelity. Additionally, it uses Inter- and Intra-Gaussian Context Models to effectively eliminate redundancy among the irregular Gaussian blobs.

3.1.5 Structured Compression. **Structured Compression** addresses the compression inefficiency of the original 3DGS [42], where existing compression techniques [1] achieve poor efficiency due to difficulty in identifying redundancy among unstructured 3D Gaussians. These methods organize 3D Gaussians based on spatial relationships to improve compression efficiency. Specifically, Morgenstern *et al.* [68] introduce the Parallel Linear Assignment Sorting (PLAS) algorithm to organize high-dimensional Gaussian attributes into a 2D grid, grouping spatially adjacent 3D Gaussians expected to have similar attributes. Octree-GS [81] utilizes an Octree [66] structure to dynamically select Level-of-Detail (LoD) 3D Gaussians, enabling consistent real-time rendering for large-scale scenes. Papantonakis *et al.* [73] use adaptive

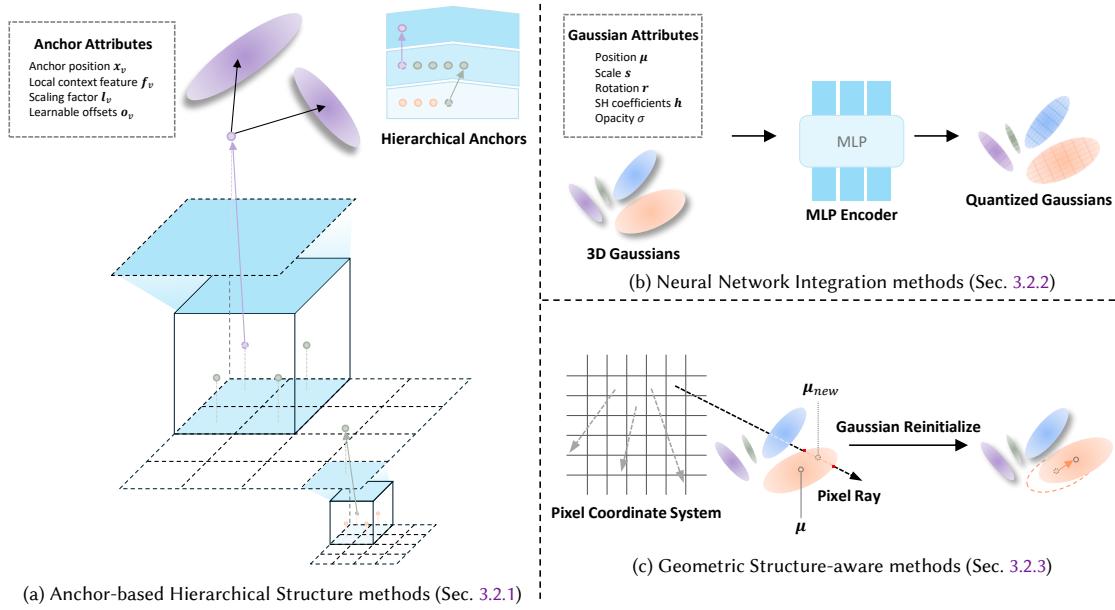


Fig. 4. Overview of Restructuring Compression strategies for Static 3DGS. These approaches fundamentally modify the 3DGS model architecture to achieve an efficient scene representation. (a) Anchor-based Hierarchical Structure methods (Sec. 3.2.1) introduce hierarchical representation to 3DGS [42] using sparse anchors. (b) Neural Network Integration methods (Sec. 3.2.2) replace 3DGS representations with neural networks. (c) **Geometric Structure-aware** methods (Sec. 3.2.3) exploit geometric properties.

voxelization by computing the world-space volume of a single pixel and using 3D Gaussian overlap frequency within this volume as a spatial redundancy metric. MesonGS [113] utilizes an Octree structure to voxelize the 3D Gaussian positions μ and uses RAHT (Region Adaptive Hierarchical Transform) to compress the Gaussian attributes in adjacent voxels. HGSC [38] compresses 3D Gaussian positions μ using an Octree structure and uses a multi-level attribute compression strategy. GoDe [84] proposes a model-agnostic scalable compression framework that constructs a multi-level 3D Gaussian hierarchy, enabling dynamic adjustment of detail and compression rates without retraining. Wang *et al.* [102] introduce an adaptive voxelization algorithm utilizing transform coding tools developed for point cloud compression [28]. GHAP [105] compresses Gaussian attributes through block-wise Gaussian Mixture Reduction (GMR) based on KD-tree [7] partitioning. HRGS [52] introduces hierarchical block-level optimization, enabling high-quality, high-resolution 3D reconstruction even with limited GPU memory.

3.2 Restructuring Compression

Restructuring Compression aims to fundamentally modify the original 3DGS [42] model architecture to obtain an efficient scene representation. This survey classifies Restructuring Compression methods into three main strategies:

- **Anchor-based Hierarchical Structure** methods: This strategy addresses the lack of a hierarchical representation in the original 3DGS [42] by utilizing sparse anchor representations.
- **Neural Network Integration** methods: This strategy achieves an efficient scene representation by replacing the original 3DGS representations with neural networks.

- **Geometric Structure-aware** methods: This strategy achieves an efficient scene representation by utilizing scene geometric properties.

3.2.1 Anchor-based Hierarchical Structure Methods. **Anchor-based Hierarchical Structure** methods utilize the sparse anchor representations derived from the voxelization process as the structural foundation. Scaffold-GS [62] proposes a hierarchical scene representation based on anchors to address the lack of a hierarchical structure in the original 3DGS [42], which results in excessive redundant 3D Gaussians. The framework uses the point clouds $P \in \mathbb{R}^{N \times 3}$ to voxelize the scene as follows: $V = \{\text{Floor}(|\frac{P}{\epsilon}|)\} \cdot \epsilon$, where ϵ is a value that represents the size of the voxel V , and $\text{Floor}(\cdot)$ is the floor function, which rounds down to the nearest smaller integer. Each voxel V serves as an anchor $x_v \in V$, which is assigned the following attributes: local context features $f_v \in \mathbb{R}^{32}$, scaling factors $l_v \in \mathbb{R}^3$, and k learnable offsets $O_v \in \mathbb{R}^{k \times 3}$. The framework produces k neural Gaussians from each anchor x_v as follows:

$$\{\mu_0, \dots, \mu_{k-1}\} = x_v + \{O_0, \dots, O_{k-1}\} \cdot l_v. \quad (23)$$

Subsequently, the anchor attributes $\{a_0, \dots, a_{k-1}\}$ of each neural Gaussian are predicted using an MLP F_a . The MLP takes the view-dependent features \hat{f}_v , the relative camera-anchor distance δ_{vc} , and the view direction d_{vc} as input, as follows:

$$\{a_0, \dots, a_{k-1}\} = F_a \left(\hat{f}_v, \delta_{vc}, d_{vc} \right). \quad (24)$$

The view-dependent features \hat{f}_v is a weighted sum of multi-resolution local context features, where the weights depend on the relative distance δ_{vc} and the view direction d_{vc} . This approach enables the prediction of anchor attributes to be robust to changes in resolution and view direction. HAC [14] uses hash features derived from the interpolation of each anchor in a hash grid as contexts to estimate the probability of each quantized anchor attribute, enabling efficient entropy coding. CompGS [60] uses a hierarchical hybrid primitive structure that employs a small set of anchor primitives to predict the other primitives. ContextGS [107] divides the anchors into multiple hierarchical levels using a bottom-up voxelization strategy, as shown in Fig. 4. Subsequently, it introduces an autoregressive model that effectively predicts undecoded anchors by utilizing the already decoded anchors from the lower levels. HEMGS [58] achieves hybrid lossy-lossless compression of anchor attributes using a Hybrid Entropy Model (HEM). The framework incorporates a variable-rate predictor for lossy compression and combines a hyperprior network with an autoregressive network for improved lossless entropy coding. CAT-3DGS [121] utilizes a Triplane-based hyperprior and a Spatial Autoregressive Model (SARM) to leverage the spatial inter-correlation among Gaussian primitives. It introduces a Channel-wise Autoregressive Model (CARM) to utilize the channel-wise intra-correlation within individual primitives. PCGS [12] uses progressive masking to increase anchor quantity and progressive quantization with level-wise context modeling to refine anchor quality, enabling progressive bitstream generation. TC-GS [106] uses KNN to predict the distribution of Gaussian attributes based on the Tri-plane. SHTC [114] uses a Karhunen-Loeve Transform (KLT)-based layer for data decorrelation and a sparsity-guided enhancement layer for residual compression.

3.2.2 Neural Network Integration Methods. **Neural Network Integration** methods modify the original 3DGS [42] model architecture by replacing the original 3DGS representations with neural networks. This strategy utilizes the generative capacity of neural networks to compress Gaussian attributes. EAGLES [26] introduces an MLP decoder, denoted as $D : \mathbb{Z}^l \rightarrow \mathbb{R}^k$, which serves to decode the quantized latent vectors into actual attributes \mathcal{A} . As shown in Fig. 4, the framework projects the uncompressed attributes $a \in \mathcal{A}$ into the latent space. This projection applies the inverse decoder function D^{-1} to obtain $\hat{q} = D^{-1}(a)$. The framework rounds \hat{q} to the nearest integer \bar{q} and reconstructs

the attributes as $\mathbf{a} = D(\bar{\mathbf{q}})$. NeuralGS [93] performs clustering based on the similarity of Gaussian attributes and assigns a small MLP to each cluster to encode the Gaussian attributes. Liu *et al.* [59] introduce the Mixture of Priors (MoP) and Coarse-to-Fine Quantization (C2FQ) strategies. The MoP strategy uses multiple lightweight MLPs with a gating mechanism to generate diverse prior features, improving conditional entropy modeling accuracy.

3.2.3 Geometric Structure-aware Methods. **Geometric Structure-aware** methods modify the original 3DGS [42] model architecture to utilize the geometric properties of the scenes. This approach achieves a more efficient representation with high visual fidelity by incorporating geometric priors into a scene representation. SAGS [100] applies curvature-aware densification to COLMAP [85] point clouds to populate under-represented areas. It then introduces a structure-aware encoder that processes these densified points, utilizing Graph Neural Networks (GNNs) on a local-global graph representation to learn robust structural features. Mini-Splatting [23] reconstructs the spatial distribution of 3D Gaussians through densification with Blur Split and Depth Reinitialization. Specifically, Depth Reinitialization determines the Gaussian depth as the mid-point of the two intersection points between the ray of each pixel and a 3D Gaussian, as shown in Fig. 4. These new points are reprojected into world space, contributing to the improvement of the inefficient spatial distribution. 3D-HGS [53] addresses the limitations of the original 3DGS [42] in modeling shape and color discontinuities by introducing a novel 3D Half-Gaussian (3D-HGS) kernel. The kernel efficiently captures higher frequency information by using different opacity values for each half.

4 DYNAMIC

As 3DGS [42] has shown remarkable success in static scene representation and real-time rendering, a natural yet challenging extension lies in modeling dynamic scenes. Real-world environments often contain non-rigid motions, occlusions, and temporally varying geometry and appearance. However, most early 3DGS methods are designed under the static scene assumption, which makes them unsuitable for applications such as video-based rendering, AR/VR, or autonomous driving.

To bridge this gap, recent approaches [57, 63] have extended the 3DGS framework to handle temporal dynamics. The primary challenge in this domain is achieving a compact and efficient representation that captures temporal variations while keeping computational and memory overhead low. Specifically, dynamic scenes demand additional storage to represent temporal variations across time [47], which scales linearly or exponentially depending on the granularity and duration of motion being modeled.

To address this, existing methods that address dynamic 3DGS typically fall into two broad categories. Following the same taxonomy introduced in the static setting, we analyze dynamic 3DGS methods through the lens of two complementary directions: **(1) Parameter Compression** and **(2) Restructuring Compression**. This dichotomy serves as a unified framework in our survey to assess how efficiently different approaches represent both static and dynamic scenes under a common goal of reducing redundancy:

- **Parameter Compression:** These approaches focus on reducing redundancy in the spatio-temporal domain using pruning, quantization, and entropy-based methods, independent of the model architecture.
- **Restructuring Compression:** These approaches reduce representation size by reusing shared structures across frames through architectural designs such as anchor-based modeling or learned deformation fields.

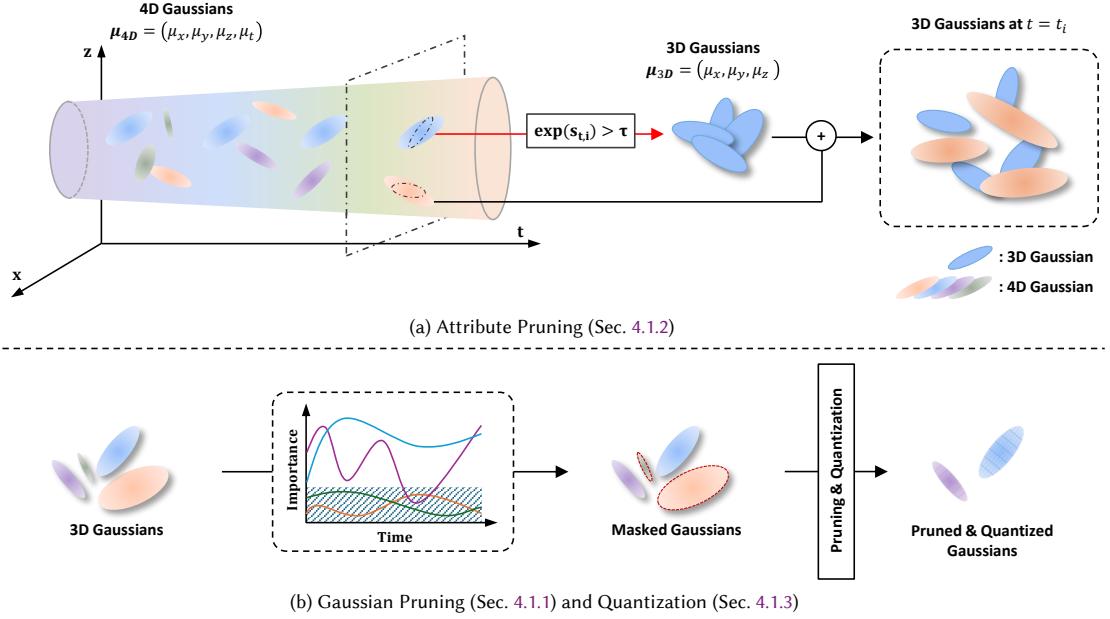


Fig. 5. Overview of Parameter Compression strategies for Dynamic 3DGs. These approaches reduce redundancy in the Gaussian representation without modifying the rendering architecture. (a) **Attribute Pruning** (Sec. 4.1.2) removes temporally inactive components from 4D Gaussians, reducing them to 3D Gaussians that capture the invariant spatial attributes. The pruned 3D Gaussians are then combined with time-dependent 4D Gaussians at specific timestamps to reconstruct dynamic scenes while preserving spatial fidelity. (b) **Gaussian Pruning** (Sec. 4.1.1) discards less-contributing Gaussians based on temporal importance, followed by **Quantization** (Sec. 4.1.3), which discretizes Gaussian parameters to achieve compactness while preserving rendering quality.

4.1 Parameter Compression

Parameter compression methods aim to reduce redundancy in the Gaussian representation without modifying the core rendering architecture. These techniques typically operate at the level of individual Gaussians or their attributes, making them lightweight and easily integrated into existing 3DGs pipelines. We categorize recent parameter compression approaches as the following subtypes:

- **Gaussian Pruning:** This strategy eliminates low-contributing Gaussians based on temporal activity.
- **Attribute Pruning:** This strategy removes low-impact Gaussian attributes.
- **Quantization:** This strategy discretizes Gaussian parameters.
- **Entropy-based:** This strategy exploits entropy in dynamic 3DGs.

These methods serve as effective post-processing or training-time tools to trim over-parameterized models while maintaining high rendering quality, as summarized in Fig. 5.

4.1.1 Gaussian Pruning. **Gaussian pruning** strategies have been widely explored in the field of static 3DGs to remove Gaussians with minimal visual impact [22, 25, 49, 71]. These approaches typically assess importance based on spatial metrics or image-space contributions, which refers to the projected influence of a 3D Gaussian on the 2D image plane from a given camera viewpoint. However, extending such methods to dynamic scenes introduces additional

challenges [40, 50]. For instance, the pruning method, Lee *et al.* [49] described in [static section](#) utilizes Eq. (13) and (14) to prune unnecessary Gaussians. Subsequently, a simple loss function (15) is employed to encourage the mask parameter m_n to be as small as possible, thereby enabling effective pruning. In dynamic scenes, as Gaussians move over time, the importance of individual Gaussians can also change. This method does not account for temporal variations, which can lead to a Gaussian deemed important in one frame suddenly becoming unnecessary in the next, causing rendering quality degradation or unstable pruning. To address this, TC3DGS [40] introduces a time attribute t and defines an additional mask consistency loss function:

$$\mathcal{L}_{mc} = \sum_{n=1}^N |m_{n,t} - sg(m_{n,t-1})|. \quad (25)$$

This \mathcal{L}_{mc} forces $m_{n,t}$ to remain similar to $m_{n,t-1}$, thereby ensuring that the importance of the Gaussians is learned in a temporally stable manner. This loss is then added to the basic loss (15) to implement the final loss function. This allows the model to better grasp the global importance of Gaussians and effectively prune unnecessary ones, maximizing storage space and rendering efficiency.

Building upon the foundation described in [static section](#), Speedy-Splat [32] computes a per-Gaussian sensitivity score by aggregating gradients across all static camera poses ϕ . SpeeDe3DGS [97], designed for dynamic scenes, extends this idea with the Temporal Sensitivity Pruning Score. Since Gaussians deform and change their contributions over time t , it computes time-dependent gradients for the rendered image $I_{\mathcal{G}_t}(\phi)$ and aggregates them across both camera poses and timestamps, enabling more stable and effective pruning over the dynamic sequence. In this direction, 4DGS-1K [120] further advances pruning by jointly evaluating temporal and spatial scores. In particular, its temporal score quantifies the lifespan of each Gaussian along the time axis. By modeling the temporal opacity function $p_i(t)$ and examining its second derivative, the method captures how steadily a Gaussian persists over time or how abruptly it appears or disappears, enabling more informed temporal pruning.

Also, Ex4DGS [47] introduces a pruning strategy that leverages point backtracking, which traces image-space errors back to the responsible Gaussians. After computing pixel-wise errors by comparing rendered and GT images, the method exploits the backward pass to estimate each Gaussian’s contribution to the overall error. These contributions are weighted by opacity and accumulated transmittance to reflect their actual impact on pixels, and then averaged across all training views to obtain a global error threshold \mathcal{E}_{total} . Pruning is performed at predefined steps, where Gaussians with errors exceeding \mathcal{E}_{total} are removed.

Another line of pruning-based strategies focuses on handling newly appearing objects while maintaining memory efficiency. 3DGStream [90] assigns new Gaussians to emerging objects and applies Adaptive 3DG Quantity Control. When the view-space positional gradient, which measures how sensitively pixel colors change with respect to a Gaussian’s 2D projection, exceeds a threshold, new Gaussians are created, whereas those with gradients below the threshold are discarded. This allows the model to flexibly represent new content while preventing uncontrolled growth. Complementarily, Instant4D [64] reduces redundancy and mitigates self-occlusion by partitioning the world space into a regular voxel grid and retaining only the centroid within each occupied voxel, keeping the representation compact during dynamic updates.

4.1.2 Attribute Pruning. **Attribute pruning** focuses on removing specific attributes or parameters of Gaussians that are redundant or have minimal impact on the scene representation. A notable example is Hybrid 3D-4DGS [72], which prunes time attributes t from the 4D Gaussians. The core idea of Hybrid 3D-4DGS is a hybrid representation that models

a scene by decomposing it into static and dynamic regions. It begins with a complete 4D Gaussian representation and then identifies Gaussians that do not change over time. These static 4D Gaussians are converted into 3D Gaussians, which effectively removes their temporal dimension parameters. This significantly reduces the total number of parameters, leading to lower memory consumption and faster training. Unlike prior works that often identify static and dynamic content by analyzing the flow of Gaussians [47, 61, 69], Hybrid 3D-4DGS leverages a 4D coordinate system. It uses a time-axis scale parameter, $\exp(s_{t,i})$, for each Gaussian. If this temporal scale exceeds a predefined threshold τ , the Gaussian is classified as static and its time-attributes are pruned.

4.1.3 Quantization. **Quantization** is a core parameter compression technique that converts continuous, high-precision values into a discrete, low-precision representation. Vector Quantization (VQ) is a specific and powerful form of quantization that exploits the inherent redundancy in 3DGS parameters. It groups similar parameters into a few clusters, stores the cluster centers in a codebook, and then replaces the original parameters with the index of their corresponding cluster. This process significantly reduces storage, as millions of Gaussians can be represented with just the codebook and a set of indices. VQ is a widely used parameter compression technique in various applications, including deep network compression [16, 27] and generative models [30, 79, 99]. This demonstrates its proven efficacy in handling large-scale, redundant data, which is precisely the challenge posed by 3DGS. While initially developed for static scenes [22, 49, 69, 71], VQ is crucial for dynamic 3DGS due to the massive storage requirements of temporal data. The core principle remains the same, but the challenge lies in applying it to dynamic parameters, such as temporal deformation fields.

Not all Gaussian parameters equally affect rendering quality, making uniform quantization inefficient. Sensitivity-based quantization allocates higher bit precision to more critical parameters and lower precision to less important ones. In dynamic 3DGS, some parameters are more sensitive to temporal changes. TC3DGS [40] addresses this with a gradient-aware mixed-precision scheme that measures each parameter’s sensitivity via its gradient magnitude. Parameters with larger gradients receive higher precision, while less impactful ones are quantized with fewer bits, balancing compression and reconstruction accuracy.

4.1.4 Entropy-based. The concept of **entropy** is leveraged in dynamic 3DGS in two ways. The first is as a compression codec used in the final compression stage after model optimization. The second is as a regularization loss during the training process to induce model sparsity. These two approaches stem from fundamentally different philosophies, and it’s crucial to understand the distinct advantages and disadvantages of each.

Entropy is first used as a compression codec in the final compression stage. A key challenge with 4D Gaussians is that each frame consumes storage comparable to a keyframe, causing high memory usage for long sequences. HiFi4G [41] addresses this with residual compensation, quantization, and entropy encoding. It exploits the small differences between adjacent frames by computing residuals between each Gaussian and its keyframe attributes, which are often near zero. These residuals are quantized and encoded using a Ranged Arithmetic Numerical System (RANS) [20], which efficiently compresses skewed frequency distributions. The encoded integer stream can be decoded to reconstruct the original attributes, enabling HiFi4G to achieve real-time compression and decompression even for long sequences.

The use of entropy as a regularization loss can be seen in MEGA [125], which introduces an entropy-constrained Gaussian deformation technique to enhance the utilization of each Gaussian and reduce the total number required. Conventional Yang *et al.*’s 4DGS method [119] often assumes that 4D Gaussians exhibit only linear motion over time with constant covariance. Additionally, temporal decay opacity ensures that each Gaussian is only visible during a specific time, with less than 10% of Gaussians participating in the rendering at any given moment. To overcome these

limitations, MEGA improves the flexibility of Gaussian motion and geometric structure. It then introduces a spatial opacity-based entropy loss to encourage each Gaussian’s spatial opacity σ to be close to 0 or 1:

$$\mathcal{L}_{opa} = \frac{1}{N} \sum_{j=1}^N (-\sigma_j \log(\sigma_j)). \quad (26)$$

Periodically during the optimization process, Gaussians with near-zero opacity are aggressively pruned, as they are considered to not be contributing to the scene’s representation. This reduces the model’s memory footprint and increases computational efficiency.

4.2 Restructuring Compression

While parameter compression methods reduce redundancy at the level of individual Gaussians or their attributes, restructuring compression techniques aim to achieve a more efficient representation by modifying the underlying architecture itself. These methods are **architecture-dependent** in that they introduce additional structural priors or modules that enable more efficient modeling of dynamic scenes. Specifically, recent restructuring compression methods can be broadly categorized into three major approaches:

- **Anchor-based Representation:** This strategy compresses dynamic content using representative keyframes.
- **Canonical Deformable Representation:** This strategy models dynamic scenes by establishing a canonical space and a deformation field.
- **LoD Representation:** This strategy organizes Gaussians into multi-resolution hierarchies.

These strategies reduce parameters, improve temporal consistency, and provide inductive biases that enhance generalization in dynamic environments, as summarized in Fig. 6. In the following subsections, we review representative methods in each category. While we group them for clarity, recent works [45, 127] often combine multiple techniques. Many state-of-the-art methods adopt hybrid designs that leverage the strengths of different paradigms, so these categories should be seen as conceptual guides rather than strict boundaries.

4.2.1 Anchor-based Representation. Recent studies [14, 60, 62, 107] have successfully introduced more compact 3DGS frameworks using anchor-based representations. This approach assigns implicit features to sparse anchor points, which are then used to predict the attributes for a broader set of neighboring 3D Gaussians. Building on this natural progression, anchor-based frameworks for static scenes are being actively extended to dynamic environments to address the significant storage overhead caused by the temporal dimension.

Anchor-based approach represents a dynamic scene’s complex deformations by using a small number of hierarchical control points or anchors to represent a large number of Gaussians. These control points increase in density in areas with complex motion, allowing the model to effectively capture even subtle non-rigid deformations [101]. MoSca [51] separates the 3D geometry and motion of a dynamic scene using a representation called 4D Motion Scaffolds. These scaffolds compactly encode the underlying deformations of the scene, while 3D Gaussians are anchored to them to represent the appearance. This enables the model to globally encode information from all video frames into a single, consistent representation. EDGS [44] models dynamic scenes efficiently through a sparse anchor-grid representation. This approach decomposes time-invariant and time-variant properties. In particular, it employs an unsupervised learning strategy to effectively filter out anchors corresponding to static regions, while querying time-variant attributes by feeding only anchors related to dynamic objects into an MLP. Another notable method is MoDec-GS [45], introducing Global-to-Local Motion Decomposition (GLMD) to capture both global and local movements. This approach extends

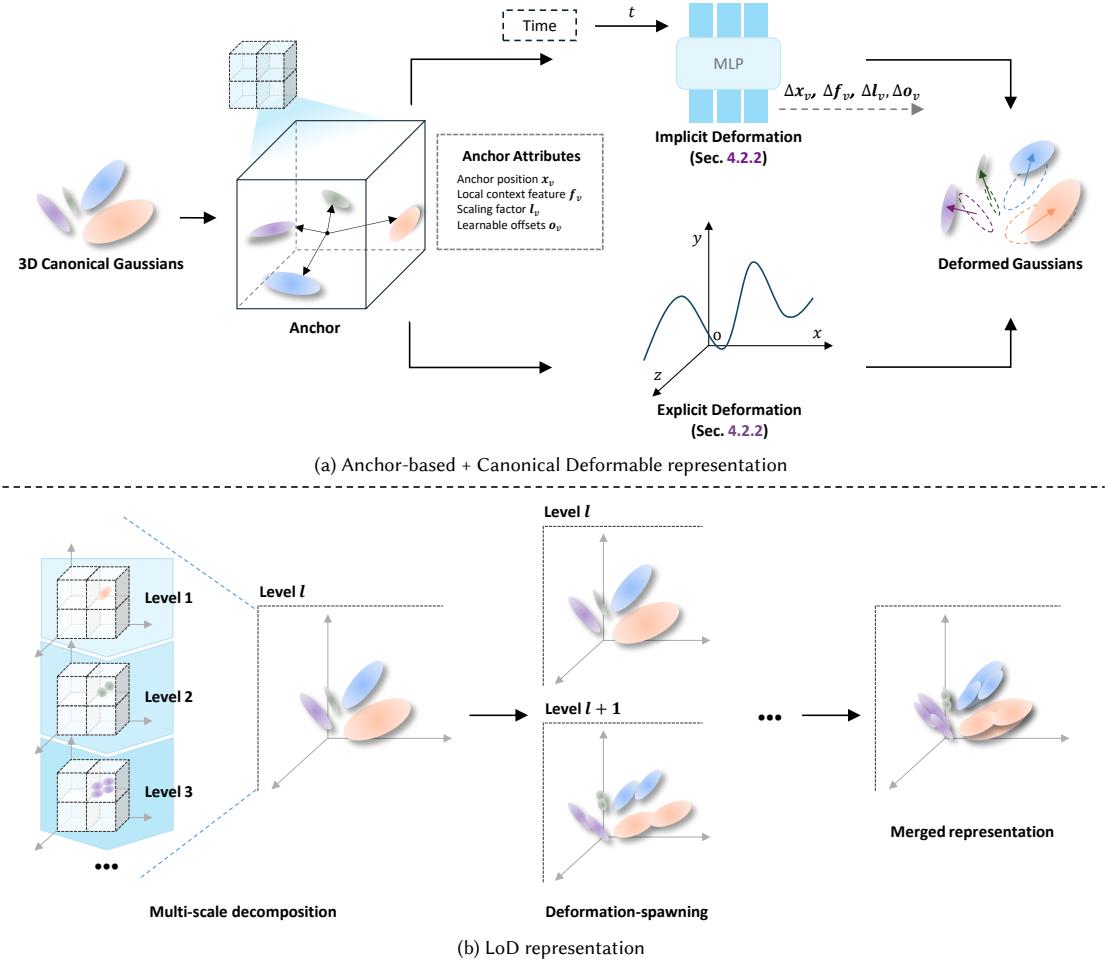


Fig. 6. Overview of Restructuring Compression strategies for Dynamic 3DGS. These approaches achieve compact yet expressive representations by restructuring the underlying 3DGS architecture to better encode temporal and spatial variations. (a) **Anchor-based and Canonical Deformable Representation** (Sec. 4.2.1, Sec. 4.2.2) combines anchor-based referencing with a canonical-deformable formulation. A small set of anchor Gaussians serve as spatial references, where each anchor encodes local context attributes and deformation parameters that map the canonical static space to dynamic frames over time. (b) **LoD Representation** (Sec. 4.2.3) organizes Gaussians into a multi-resolution hierarchy, progressively refining spatial detail through level-wise deformation and spawning

static scaffold representations to dynamic video reconstruction by using Global Canonical Scaffolds and Local Canonical Scaffolds. HAIF-GS [11] employs sparse motion anchors as deformation units to reduce redundant computation and improve efficiency. Its Hierarchical Anchors Densification (HAD) adaptively refines anchor resolution in motion-complex regions for fine-grained deformations. An Anchor Filter predicts dynamic confidence scores to suppress redundant updates in static areas, while the Induced Flow-Guided Deformation (IFGD) module aggregates multi-frame features to induce scene flow in a self-supervised manner, regularizing anchor transformations.

4.2.2 Canonical Deformable Representation. This approach represents a dynamic scene as a set of static 3D Gaussians defined in a canonical space and a deformation field that maps points from this canonical space to the time-dependent real-world by implicitly or explicitly. Efficiency is achieved by inducing dynamic changes through the deformation field, rather than storing all Gaussians for every timestamp [63].

Implicit Deformation learns time-dependent deformations from the canonical space using MLP or grid-based architectures. Instead of defining a rigid, explicit function, the deformation field is modeled as a neural network that infers the transformation for each Gaussian at a given time. This allows for the capture of complex, non-rigid movements with high fidelity. One of the early works in this field, Deformable 3DGS [118] introduces an implicit deformation field modeled by an MLP network, which, at each timestamp, takes a 3D Gaussian’s position and the corresponding time t as input to predict offsets for its position μ , orientation r , and scale s . Wu *et al.*’s 4DGS [109] takes a different approach by maintaining a single set of canonical 3D Gaussians and transforming them at each timestamp via a Gaussian deformation field. This field, comprising a temporal-spatial structure encoder and a multi-head Gaussian deformation decoder, leverages a multi-resolution HexPlane [9], a representation for 4D volumes that uses six planes of learned features to efficiently compute features for spatio-temporal points, thereby efficiently modeling Gaussian motion and shape changes. Building upon these, DeformGS [21] focuses on dynamic 3D tracking using multiple cameras, employing a Neural-voxel Encoding combined with an MLP to infer a Gaussian’s position, rotation, and even a shadow scalar. This scalar is a value between 0 and 1 that represents the shadow intensity. This approach further incorporates regularization terms based on conservation of momentum and isometry to reduce trajectory errors.

Explicit Deformation models dynamic scenes using mathematically predefined functions to represent motion trajectories and temporal changes. Instead of relying on a neural network to predict transformations, this approach directly parameterizes motion with a set of explicit variables, which can include polynomial coefficients or keyframe-based interpolators. This strategy often leads to more compact representations by avoiding the need for a deep neural network. For example, Li *et al.* [57] model dynamic scenes using an explicit deformation approach that achieves compactness through low-order polynomial functions. The position μ and orientation r of each Gaussian are represented by these polynomials, with motion trajectories inspired by [31, 42] and rotations by [36, 39]. Representing time-varying motion with a small set of polynomial coefficients greatly reduces parameters compared to per-frame modeling, making the parameter count proportional to the number of Gaussians rather than frames. While other techniques like splatted feature rendering aid compactness, the core efficiency of Li *et al.*’s method lies in its explicit polynomial representation of dynamic motion. Another approach that uses explicit deformation is Ex4DGS [47], which efficiently represents the motion of dynamic 3DGS by employing keyframe-based interpolation. Instead of storing motion information for every frames, Ex4DGS explicitly stores the position and rotation of Gaussians only at a small number of keyframes. The scene’s motion at any given timestamp is then synthesized by interpolating between adjacent keyframes. For the Gaussian’s position μ , it uses Cubic Hermite Spline (CHip) [5], which provides a smooth and continuous interpolation

$$\mu(t) = \text{CHip}(p_n, m_n, p_{n+1}, m_{n+1}; t'). \quad (27)$$

Here, p_n is the position at the n -th keyframe and m_n is its tangent vector, which is calculated based on the position change between keyframes to avoid extra storage. For the Gaussian’s rotation, it uses Spherical Linear Interpolation (Slerp) [87] to ensure a consistent and bias-free interpolation of the quaternion q ,

$$q(t) = \text{Slerp}(r_n, r_{n+1}; t'), \quad (28)$$

where r_n is the rotation at the n -th keyframe. By using these explicit interpolators and keyframes, Ex4DGS avoids the need to store information for every single timestamp, which significantly reduces the model size. This approach effectively balances expressiveness for complex motion with the practicality of minimal memory overhead.

4.2.3 LoD Representation. As we described in the static section, **LoD representation** contributes to compact and efficient 3DGS frameworks by dynamically adjusting the level of detail of objects based on the viewer’s perspective, distance, and importance. This makes it valuable for various applications like gaming [8, 91] and virtual reality [86, 123]. Inspired by scalable video encoding [29, 67], Scale-GS [117] employs LoD techniques with deformation and spawning to build a compact and efficient dynamic 3DGS framework. It introduces a multi-scale Gaussian representation, where large Gaussians capture coarse structures and small ones refine high-frequency details, improving efficiency by avoiding unnecessary optimization. Scale-GS hierarchically partitions the scale space through recursive binary splitting. From the initial frame, it estimates each Gaussian’s scale and defines the maximum, minimum, and mean scales ($s_{\min}^{(l)}, s_{\max}^{(l)}, s_{\text{mean}}^{(l)}$) for each level l . If a finer level $l+1$ is required, the scale range is divided as:

$$[s_{\min}^{(l)}, s_{\max}^{(l)}] \leftarrow [s_{\text{mean}}^{(l)}, s_{\max}^{(l)}], \quad (29)$$

$$[s_{\min}^{(l+1)}, s_{\max}^{(l+1)}] \leftarrow [s_{\min}^{(l)}, s_{\text{mean}}^{(l)}]. \quad (30)$$

This hierarchy enables coarse-to-fine optimization, where large-scale Gaussians first approximate the global structure using low-resolution views, and smaller ones are later activated to refine fine details, greatly reducing redundant computation and training time. A similar hierarchical principle appears in 4DGCP [126], which decouples motion into two levels: a rigid transformation capturing large object movements, and a residual deformation that models non-rigid shape changes and fine-grained local dynamics. This decomposition mirrors a coarse-to-fine strategy, allowing the system to first account for dominant global motion and then refine complex deformations efficiently.

5 DATASETS AND EVALUATION

Evaluating efficient 3DGS and its dynamic extensions requires efficiency-oriented metrics beyond conventional rendering quality measures. While earlier studies focused primarily on visual quality (PSNR, SSIM [108], LPIPS [124]), efficient approaches target a quality-efficiency trade-off. We summarize commonly used datasets and metrics from the literature and analyze the balance between visual fidelity and efficiency.

5.1 Datasets

5.1.1 Static Scene Datasets. As static scene reconstruction and novel view synthesis have established themselves as fundamental research areas in computer vision and graphics, numerous high-quality datasets have been developed to support and evaluate various methodologies. In this section, we review five representative static scene datasets widely used for 3D reconstruction. Each dataset presents unique challenges for evaluating model performance.

Tanks and Temples (TNT) [43] introduces a benchmark dataset for evaluating large-scale 3D reconstruction techniques. The dataset comprises real-world captures acquired using an industrial laser scanner at submillimeter accuracy. The dataset captures videos primarily using gimbal-stabilized cameras (DJI Zenmuse X5R and Sony a7S II) at 4K resolution. TNT includes 14 diverse scenes with 100–400 views each, categorized into intermediate (8 outdoor) and advanced (4 indoor, 2 outdoor) groups, with images extracted at 1920×1080 resolution.

Deep Blending [34] introduces a comprehensive dataset of 2,630 real photographs from 19 scenes. The dataset comprises captures from Chaurasia *et al.* [10] (7 scenes), Hedman *et al.* [35] (4 scenes), the Eth3D benchmark (1 scene),

	TNT [43]	Deep Blending [34]	NeRF-Synthetic [6]	BungeeNeRF [111]	Mip-NeRF 360 [4]
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
Link					

Table 1. **Summary of representative static 3DGS datasets.** Type indicates whether scenes are real-world or synthetic. Modality specifies the camera configuration (monocular vs. multi-view). #Views denotes the number of viewpoints per scene. #Scenes represents the total number of scenes, where red indicates outdoor and blue indicates indoor scenes. Resolution indicates image dimensions in pixels (width × height). Environment describes the scene setting (indoor vs. outdoor).

and newly captured scenes (7 scenes). Each scene contains 12–418 images from different viewpoints, spanning 5 indoor and 14 outdoor environments with resolutions ranging from 1228 × 816 to 2592 × 1944 pixels.

NeRF-Synthetic [6] presents a synthetic dataset of eight path-traced objects designed to evaluate novel view synthesis on complex geometry and realistic non-Lambertian materials. The dataset employs the Blender Cycles renderer for high-fidelity physically-based rendering. Each scene provides 100 training views and 200 test views. The eight objects (Chair, Drums, Ficus, Hotdog, Lego, Materials, Microphone, and Ship) exhibit complex geometry and intricate appearance properties. All scenes feature object-centric indoor setups in controlled studio environments with neutral backgrounds, rendered at 800 × 800 pixel resolution.

BungeeNeRF [111] presents multi-scale datasets for novel view synthesis in extreme multi-scale scenarios. The collection includes synthetic data from Google Earth Studio [6] and Blender, and real-world UAV captures. The dataset contains twelve outdoor cities (New York, San Francisco, Sydney, Seattle, Chicago, Quebec, Amsterdam, Barcelona, Rome, Los Angeles, Bilbao, and Paris), plus additional landscape and Blender-synthetic environments. Each city scene provides approximately 220–463 multi-viewpoint images along the camera trajectory.

Mip-NeRF360 [4] introduces a real-world dataset for evaluating novel view synthesis in unbounded 360-degree scenes. The dataset captures 9 scenes with 100–330 images per scene. The dataset comprises images captured using a Sony NEX C-3 for 5 outdoor scenes (bicycle, flowers, garden, stump, treehill) and a Fujifilm X100V for 4 indoor scenes (room, counter, kitchen, bonsai), with an average image resolution of 4946 × 3286 pixels.

5.1.2 Dynamic Scene Datasets. As dynamic 3DGS has become a popular research area, a wide variety of datasets have been introduced to support the rapid pace of development. In this section, we will review five representative datasets that are commonly used across the dynamic 3DGS literature previously discussed. These datasets are frequently cited as benchmarks for evaluating the performance of new methods.

Technicolor Dataset [83] is a multi-view light-field video collection of various dynamic scenes, including close-ups of human faces and animated objects. It is captured using a synchronized 4 × 4 camera grid at 30 frames per second with a high resolution of 2048 × 1088 pixels. The dataset’s precise synchronization and calibration make it a strong benchmark for validating fine details and complex textures in dynamic 3DGS methods. However, a key consideration is that each sequence has unique shift and calibration tables, requiring a specific post-processing pipeline for proper use.

D-NeRF [76] is a synthetic extension of a static NeRF [6] benchmark, designed for dynamic scenes. It features eight scenes with large deformation and non-Lambertian materials, which are surface whose color and brightness

	Technicolor [83]	D-NeRF [76]	HyperNeRF [74]	N3DV [56]	NeRF-DS [116]
Venue	CVPR'17	CVPR'21	SIGGRAPH Asia'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
Environment	Indoor	Mixed	Indoor	Indoor	Indoor
Link					

Table 2. **Summary of representative dynamic 3DGS datasets.** Type indicates whether the dataset contains real-world captured scenes or synthetically generated scenes (Real vs. Synthetic). Modality specifies the camera configuration used for data acquisition (Monocular vs. Multi-view). #Views denotes the number of camera viewpoints available per scene in the dataset. #Scenes represents the total number of distinct scenes included in the dataset, where **red numbers** indicate outdoor scenes and **blue numbers** indicate indoor scenes. #Frames represents the number of frames per scene. Since each dataset contains scenes of varying lengths, the frame count is expressed as an approximate range. Resolution indicates the video resolution in pixels (width × height) used for each dataset. Environment describes the setting where the scenes are captured or created (Indoor vs. Outdoor).

depend on the viewing direction as well as the light’s direction. This enables the modeling of complex and realistic light interactions, such as those on glossy or reflective surfaces. The dataset is created by rendering 100–200 multi-view still frames per scene at 800 × 800 resolution and arranging them into sequences over time. Its primary advantage for 3DGS is that it is a clean dataset, free from real-world camera or geometric errors, making it ideal for evaluating an algorithm’s core dynamic deformation modeling capabilities. However, the lack of real-world noise and variations limits its use for validating generalization to real-world multi-view videos.

HyperNeRF [74] is a collection of video sequences captured with a single moving camera, specifically designed to include scene with topology changes, such as fluids, contacts, and separation, which are lacking in prior public datasets. The videos, each lasting 30–60 seconds, are subsampled to 15 fps and camera poses for all frames are estimated using COLMAP. For training, every fourth frame is used, with intermediate frames reserved for validation. From a 3DGS perspective, this dataset is valuable for testing the robustness of methods that handle topology changes and non-rigid deformations. However, its performance is sensitive to failures or ambiguities in single-camera pose registration, meaning that COLMAP’s alignment quality and the frame sampling strategy directly impact the final result.

Neural 3D Video Dataset [56] is a collection of real-world multi-camera videos that includes everyday indoor scenes, cooking, people, and challenging dynamic and optical effects like fire and steam. It also captures complex effects such as reflections, transparency, self-shadows, volumetric effects, and even topology changes like pouring liquids. This dataset is captured using 21 synchronized GoPro Black Hero 7 cameras, shooting at 2028 × 2704 resolution at 30 fps. Camera parameters are precisely estimated using COLMAP. Typically 18 views are used for training, and 1 view for qualitative and quantitative evaluations. From a 3DGS perspective, this dataset’s strength lies in its high-resolution, multi-view nature, and the inclusion of challenging dynamic and optical phenomena. This makes it highly suitable for evaluating a model’s real-world performance, as well as for comparing model compression and efficiency for long sequences. A noted limitation is the potential for color inconsistencies between views due to differences in camera color correction, which may require post-processing.

NeRF-DS [116] is a real-world collection focused on dynamic specular objects, such as moving mirrors, metal, and glossy surfaces. It consists of eight scenes from everyday environments with various types of motion and deformation. This dataset is created specifically to address the scarcity of such dynamic specular cases in existing dynamic NeRF

datasets. The data is captured using two rigidly mounted forward-facing cameras shooting simultaneously, with each scene containing two videos of approximately 500 frames. One video is used for training, and the other is used for testing. This design avoids the unrealistic "teleporting camera" problem of alternating between cameras. Camera poses are registered using COLMAP, with the stability of the SfM process improved by pre-applying a moving object mask obtained with MiVOS [15]. From a 3DGS perspective, a key advantage of this dataset is that it enables evaluation of real-world challenges such as dynamic specular highlights and reflections, foreground–background separation, and the role of masking moving foregrounds during pose registration. It has been quantitatively shown that omitting such masks leads to significant errors, underscoring pose sensitivity and reflection-induced breakdowns of multi-view consistency.

5.2 Evaluation Metrics

To evaluate efficient 3DGS and its dynamic extensions, we consider both visual fidelity metrics and efficiency-oriented indicators. The following metrics are commonly adopted in the literature:

- **PSNR (Peak Signal-to-Noise Ratio)**: PSNR quantifies the pixel-level accuracy between a rendered image and the GT reference. Higher PSNR values indicate lower reconstruction error and better fidelity.
- **SSIM (Structural Similarity Index Measure) [108]**: SSIM measures perceptual similarity by considering luminance, contrast, and structural information between two images. A higher SSIM score represents a closer resemblance to the GT from a perceptual perspective.
- **LPIPS (Learned Perceptual Image Patch Similarity) [124]**: LPIPS evaluates perceptual similarity using deep neural network features that approximate human visual judgment. Lower LPIPS values correspond to higher perceptual quality.
- **Model Size**: Model size is reported either in megabytes (MB) or in terms of the total number of Gaussians. A smaller model size indicates more compact representations and better memory efficiency.
- **Compression Ratio / Reduction Percentage**: These metrics measure the degree of compactness achieved compared to the original model. A higher compression ratio (or reduction percentage) reflects more effective elimination of redundancy while ideally preserving rendering quality.
- **Training Time**: It represents the total time required to optimize the model from initialization to convergence. Faster training time highlights the practicality of a method, particularly for large-scale or dynamic scenes.
- **Inference FPS (Frames Per Second)**: Inference speed is measured in FPS, indicating how efficiently a model can render frames in real time. Higher FPS values are crucial for interactive applications such as AR/VR and robotics.

5.3 Quantitative and Visual Comparison of Gaussian Splatting Methods

As shown in Fig. 7 and Fig. 8, we visualize representative static and dynamic 3DGS methods using bubble charts that jointly consider reconstruction quality (PSNR \uparrow , LPIPS \downarrow), rendering speed (FPS \uparrow), and model size (bubble radius). For fair and comprehensive visualization, Mip-NeRF 360 [4] is adopted for static scenes and N3DV [56] for dynamic ones, as these are the most widely used benchmarks in each category.

5.3.1 Static 3DGS. As illustrated in Fig. 7, the performance differences among static 3DGS compression methods are closely associated with the key technical features adopted by each approach. MessonGS [113], which demonstrates superior performance in the upper-left region of the chart, effectively exploits spatial adjacency through voxelization

with an Octree structure [66] and RAHT-based attribute compression. Similarly, Octree-GS [81] achieves both high visual quality and efficiency in large-scale scenes due to its hierarchical structure that enables dynamic LoD selection. Scaffold-GS [62] and SAGS [100] provide the highest visual quality due to anchor-based hierarchical representation and structural feature learning via GNNs, respectively, although these complex structures require large model sizes. ELMGS [2] and Trimming the Fat [3] maintain competitive visual quality even with extremely small model sizes due to their adoption of gradient-based aggressive pruning strategies. ContextGS [107] and HAC [14] achieve an efficient quality-size tradeoff by exploiting spatial correlations through autoregressive models and hash-based entropy coding, respectively. HEMGS [58] achieves high visual quality at low bitrates through hybrid entropy models that enable lossy-lossless compression. CAT-3DGS [121] exploits inter-channel correlations via Triplane-based hyperpriors to achieve high visual quality. EAGLES [26] and NeuralGS [94] achieve smaller model sizes through MLP decoders and clustering-based encoding, respectively, by directly utilizing the generative capabilities of neural networks for compression. Liu *et al.* [59] maintain relatively high visual quality by improving the accuracy of conditional entropy modeling through a MoP strategy. CompGS [69] and Niedermayr *et al.* [71] apply sensitivity-based k -means clustering, while Lee *et al.* [49] apply Residual Vector Quantization, both achieving simple but effective compression. LightGaussian [22] achieves effective compression of SH coefficients through a combination of knowledge distillation and VQ, enabling substantial size reduction while maintaining visual fidelity. Papantonakis *et al.* [73] and Morgenstern *et al.* [68] effectively eliminate spatial redundancy through adaptive voxelization and the PLAS algorithm, respectively, resulting in compact representations. CompGS [60] achieves efficient compression by exploiting predictive relationships between anchor and non-anchor primitives through a hierarchical hybrid primitive structure. PUP 3DGS [33] attains smaller model sizes by selectively removing 3D Gaussians with low importance through Hessian-based sensitivity scores. OMG [48] improves pruning effectiveness by performing pruning that considers local distinctiveness. SizeGS [112] achieves predictable compression ratios by explicitly modeling the hyperparameter-size relationship via a Size Estimator to guide mixed-precision quantization. GoDe [84] provides scalable quality control by enabling dynamic detail adjustment through progressive hierarchical structure construction via gradient-informed masking. PCGS [12] enables progressive decoding by supporting progressive bitstream generation through progressive masking and level-wise context modeling. SHTC [114] achieves efficient residual compression through the combination of KLT-based decorrelation and a sparsity-guided enhancement layer, maintaining high quality at reduced bitrates. FlexGaussian [95] demonstrates competitive performance without retraining due to the combination of training-free mixed-precision quantization and attribute-wise pruning, offering practical deployment advantages. RDO-Gaussian [103] maximizes compression efficiency by directly integrating rate-distortion optimization into the codeword selection process.

5.3.2 Dynamic 3DGS. As illustrated in Fig. 8, a few NeRF-based methods [54, 88] and non-compact 4DGS [119] are also included for reference, where 4D Gaussian primitives achieve higher PSNR and FPS by modeling geometry explicitly. Deformable 3DGS [118] provides one of the earliest compression strategies by avoiding the need to store all Gaussian attributes for every frame. Instead, it adopts a canonical space with a learned deformation field, yielding a substantial reduction in model size compared to per-frame representations. Building on this idea, 4DGS [109] further improves efficiency by incorporating HexPlane-based decomposed neural voxel encoding, which results in notably higher rendering speed. Subsequent works explore selective dynamic modeling. Ex4DGS [47] separates static and dynamic regions and leverages keyframe-based temporal interpolation with pruning to reduce memory footprint even further. Hybrid 3D–4DGS [72] similarly performs static–dynamic decomposition, while retaining full 4D expressiveness for dynamic Gaussians to faithfully capture complex motions. Another line of research focuses on compressing spherical

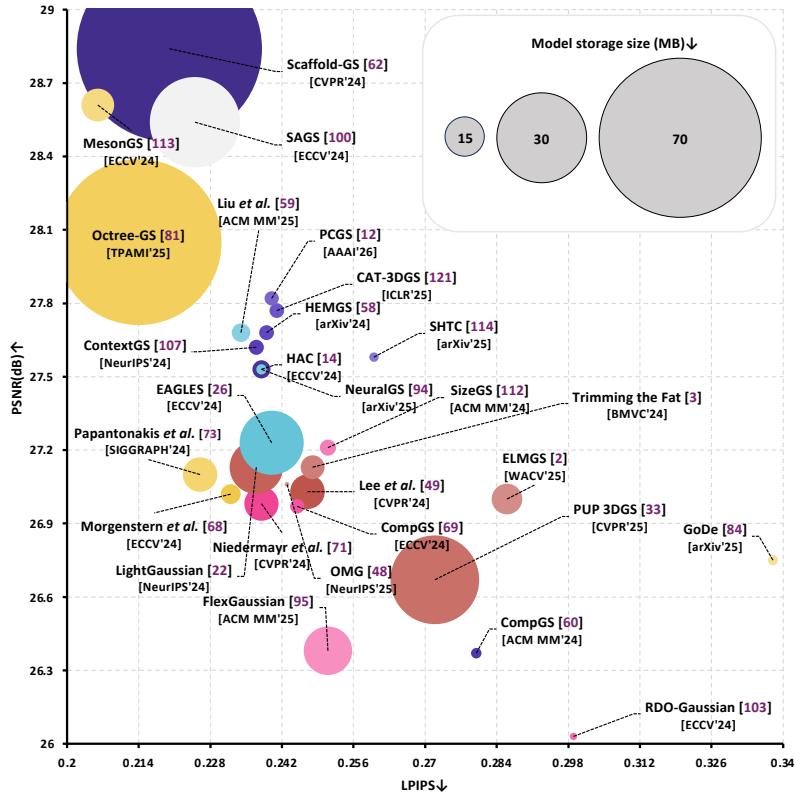


Fig. 7. Performance comparison visualization graph. Bubble plot comparing static 3DGS methods on Mip-NeRF 360 [4]. The x-axis shows perceptual quality (LPIPS ↓), the y-axis shows reconstruction quality (PSNR in dB ↑), and the bubble radius is proportional to model storage size (MB ↓). For clarity, various compactness-oriented models are distinguished by color tones: Brick red for Pruning methods (Sec. 3.1.1), raspberry for Quantization methods (Sec. 3.1.3), vivid yellow for Structured Compression methods (Sec. 3.1.5), deep violet for Anchor-based Hierarchical Structure methods (Sec. 3.2.1), sky blue for Neural Network Integration methods (Sec. 3.2.2), and light gray for Geometric Structure-aware methods (Sec. 3.2.3). Several compact designs achieve improved PSNR and lower LPIPS with reasonable storage, demonstrating the shift toward lightweight yet high-fidelity representations.

harmonics (SH), the dominant contributor to storage cost. STG [57] and MEGA [125] reduce SH complexity to obtain compact representations, with MEGA further benefiting from entropy-based regularization.

6 LIMITATIONS AND FUTURE DIRECTIONS

Efficient static 3DGS and dynamic 3DGS have overcome the issue of slow training and rendering speed inherent in NeRF, enabling real-time rendering and high-quality scene reconstruction. However, their practical deployment remains challenging due to massive memory footprint and computational overhead. A typical high-resolution static scene often contains millions of 3D Gaussians. Furthermore, 4D scene reconstruction demands even greater memory consumption because each 3D Gaussian must encode temporal information across multiple frames. To address these challenges, this survey categorizes and analyzes approaches for improving the efficiency of both static 3DGS and dynamic 3DGS. All research presented in this survey is categorized into two approaches: Parameter Compression and Restructuring Compression.

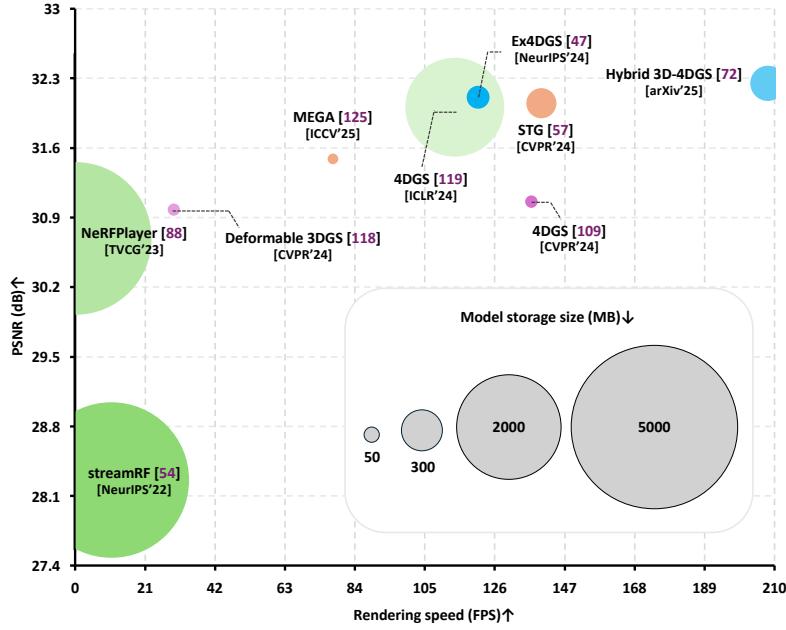


Fig. 8. Performance comparison visualization graph. Bubble plot comparing dynamic 3DGS methods on N3DV [56]. The x-axis shows rendering speed (FPS ↑), the y-axis shows reconstruction quality (PSNR in dB ↑), and the bubble radius is proportional to model storage size (MB ↓). For clarity, both compactness-oriented models (in blue tones) and non-compact baselines are shown. A clear reduction in bubble size can be observed among compact models, indicating a substantial decrease in storage cost. Moreover, several compact designs achieve improved PSNR while maintaining high rendering speed, revealing the direction of dynamic 3DGS research toward lightweight yet high-fidelity representations.

Parameter Compression approach aims to reduce redundant 3D Gaussians or their attributes without modifying the original 3DGS [42] model architecture. This makes it flexible, as it can be applied to already trained models.

For static 3DGS, parameter compression strategies can be categorized as follows:

- **Pruning** removes redundant or low-contribution 3D Gaussians.
- **Attribute Pruning** selectively removes specific components with minimal impact on rendering quality.
- **Quantization** reduces the bit precision of Gaussian attributes to minimize storage.
- **Structured Compression** organizes 3D Gaussians using spatial adjacency or hierarchical relationships to enhance compression efficiency.

For dynamic 3DGS, parameter compression strategies can be categorized as follows:

- **Gaussian Pruning** eliminates low-contributing 3D Gaussians based on temporal activity or motion magnitude.
- **Attribute Pruning** eliminates temporal parameters, effectively converting some 4D Gaussians into 3D ones.
- **Quantization** applies sensitivity-based quantization methods to temporal parameters.
- **Entropy-based Compression** employs entropy-based codecs.

Restructuring Compression fundamentally modifies the original 3DGS [42] model architecture to obtain an efficient scene representation. This category achieves compression through architectural redesign, such as hierarchical structures or alternative primitive representations.

For static 3DGS, restructuring compression strategies can be categorized as follows:

- **Anchor-based Hierarchical Method** uses sparse anchors to address the lack of hierarchical structure in the original 3DGS [42], reducing redundancy.
- **Neural Network Integration** utilizes neural networks to compress Gaussian attributes by learning compact latent representations.
- **Geometric Structure-aware Method** utilizes geometric properties of the scene to improve efficiency.

For dynamic 3DGS, restructuring compression strategies can be categorized as follows:

- **Anchor-based Representation** models complex deformations through sparse anchors.
- **Canonical Deformable Representation** maps a canonical static space to time-varying space using a deformation field.
- **LoD Representation** builds multi-resolution hierarchies to adjust detail dynamically.

Despite these advances in efficient static 3DGS and dynamic 3DGS, several critical challenges remain unresolved. This section discusses these limitations and proposes future research directions.

6.1 Hardware Optimization and Real-time Deployment

For practical deployment of static and dynamic 3DGS, optimization across diverse hardware platforms is essential. Although most current efficient 3DGS methods for both static and dynamic scenes are developed and evaluated in high-performance GPU environments, real-world applications such as AR/VR headsets, mobile devices, and edge devices operate under constrained memory and computational resources. Therefore, compression techniques and rendering pipelines that account for hardware constraints must be developed.

6.2 Long-sequence Processing for Dynamic Scenes

Most current dynamic 3DGS research targets relatively short video sequences. However, real-world applications require processing long sequences. In such long sequences, memory requirements grow exponentially, and maintaining temporal consistency becomes challenging. To this end, novel compression strategies that effectively exploit temporal redundancy are required. This can be addressed by extending keyframe-based interpolation methods to perform adaptive keyframe selection, or introducing hierarchical temporal encoding structures capable of modeling long-range temporal dependencies.

6.3 Semantically-aware Compression

Current static and dynamic 3DGS compression techniques primarily focus on pixel-level reconstruction accuracy, without sufficiently exploiting the semantic properties of scenes. Real-world scenes contain various semantic categories such as objects, backgrounds, and materials, which can be utilized to achieve more efficient compression. Specifically, scenes can be semantically segmented to allocate higher bit budgets to important foreground objects while allocating lower bit budgets to less important backgrounds. Since 3D Gaussians belonging to the same semantic category are likely to share similar characteristics, category-specific codebooks can achieve more effective compression. In dynamic scenes, explicitly separating dynamic objects from static backgrounds and applying semantically-aware compression strategies is also promising.

6.4 Generalization

Most current static and dynamic 3DGS methods employ per-scene optimization, requiring training from scratch for each individual scene. This results in substantial computational cost and training time for every new scene. In contrast, recent studies such as VGGT [104] can reconstruct 3D scenes from only a few images without per-scene optimization. However, these foundation models are computationally expensive and memory-intensive, making them impractical for resource-constrained environments such as mobile devices. Future research should focus on developing lightweight compression models that preserve the generalization capability of foundation models while being compact enough for mobile deployment. In this context, single forward pass compression frameworks without per-scene optimization, such as FCGS [13], are also promising.

6.5 User-controllable Quality-efficiency Trade-offs

Most current efficient static and dynamic 3DGS research targets fixed quality-efficiency trade-offs. However, real-world applications require dynamic adjustment of this trade-offs based on user requirements or execution environments. Therefore, flexible compression frameworks that enable diverse rate-distortion trade-offs without retraining are required. Recent studies such as GoDe [84] have begun exploring this direction, but more precise control over rate-distortion trade-offs remains challenging.

6.6 Reliability and Robustness Enhancement

For efficient static and dynamic 3DGS models to be deployed in safety-critical applications, reliability and robustness must be ensured. Specifically, adaptive compression strategies that identify safety-critical regions and allocate higher bit budgets to 3D Gaussians in those areas are required. Furthermore, quantifying compression-induced uncertainty and applying LoD to enhance details in high-uncertainty areas can improve reliability.

7 CONCLUSION

Although 3DGS enables real-time rendering through its explicit representation, its memory overhead remains one of the major challenges. Recent studies have therefore focused on designing compact and efficient representations that reduce memory usage while preserving high-fidelity scene reconstruction for both static and dynamic scenes. This survey reviews recent works that address this challenge by proposing compact and efficient 3DGS frameworks. We first introduce the core concepts of Gaussian splatting in the Preliminary section (Sec.2). Building on this foundation, the Static and Dynamic sections (Sec.3, Sec.4) systematically review recent methods that pursue compact and efficient representations under consistent criteria. We then summarize widely used datasets and evaluation metrics to support fair benchmarking across studies (Sec.5), and conclude by discussing current limitations and promising future directions in 3DGS research (Sec. 6).

References

- [1] Jyrki Alakuijala, Ruud Van Asseldonk, Sami Boukortt, Martin Bruse, Iulia-Maria Comă, Moritz Firsching, Thomas Fischbacher, Evgenii Kliuchnikov, Sebastian Gomez, Robert Obryk, et al. 2019. JPEG XL Next-generation Image Compression Architecture and Coding Tools. In *Applications of digital image processing XLII*. SPIE. <https://doi.org/10.1117/12.2529237>
- [2] Muhammad Salman Ali, Sung Ho Bae, and Enzo Tartaglione. 2025. Elmgs: Enhancing Memory and Computation Scalability through Compression for 3D Gaussian Splatting. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE. <https://doi.org/10.1109/WACV61041.2025.00257>

- [3] Muhammad Salman Ali, Maryam Qamar, Sung Ho Bae, and Enzo Tartaglione. 2024. Trimming the Fat: Efficient Compression of 3D Gaussian Splats through Pruning. In British Machine Vision Conference (BMVC). BMVA. <https://arxiv.org/abs/2406.18214f>
- [4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. <https://doi.org/10.1109/CVPR52688.2022.00539>
- [5] Richard H Bartels, John C Beatty, , and Brian A Barsky. 1987. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers Inc. <https://doi.org/doi/10.5555/35072>
- [6] Matthew Tancik Jonathan T. Barron Ravi Ramamoorthi Ben Mildenhall, Pratul P. Srinivasan and Ren Ng. 2020. Nerf: Representing Scenes as Neural Radiance Fields for View Synthesis. In European Conference on Computer Vision (ECCV). Springer, Cham. https://doi.org/10.1007/978-3-03-58452-8_24
- [7] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* (1975). <https://doi.org/10.1145/361002.361007>
- [8] David C Brogan. 2000. *Simulation Levels of Detail for Control and Animation*. Georgia Institute of Technology. <https://doi.org/doi/10.5555/931829>
- [9] Ang Cao and Justin Johnson. 2023. HexPlane: A Fast Representation for Dynamic Scenes. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. <https://doi.org/10.1109/CVPR52729.2023.00021>
- [10] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Transactions on Graphics (ToG)* (2013). <https://doi.org/10.1145/2487228.2487238>
- [11] Jianing Chen, Zehao Li, Yujun Cai, Hao Jiang, Chengxuan Qian, Juyuan Kang, Shuqin Gao, Honglong Zhao, Tianlu Mao, and Yucheng Zhang. 2025. HAIF-GS: Hierarchical and Induced Flow-Guided Gaussian Splatting for Dynamic Scene. In Advances in Neural Information Processing Systems (NeurIPS). Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2506.09518>
- [12] Yihang Chen, Mengyao Li, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2025. PCGS: Progressive Compression of 3D Gaussian Splatting. *AAAI Conference on Artificial Intelligence (AAAI)* (2025). <https://arxiv.org/abs/2503.08511>
- [13] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2024. Fast Feedforward 3D Gaussian Splatting Compression. In International Conference on Learning Representations (ICLR). <https://arxiv.org/abs/2410.08017>
- [14] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2024. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In European Conference on Computer Vision (ECCV). Springer, Cham. https://doi.org/10.1007/978-3-031-72667-5_24
- [15] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. 2021. Modular Interactive Video Object Segmentation: Interaction-to-Mask, Propagation and Difference-Aware Fusion. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. <https://doi.org/10.1109/CVPR46437.2021.00551>
- [16] Minsik Cho, Keivan A. Vahid, Qichen Fu, Saurabh Adya, Carlo C Del Mundo, Mohammad Rastegari, Devang Naik, and Peter Zatloukal. 2024. eDKM: An Efficient and Accurate Train-time Weight Clustering for Large Language Models. *IEEE Computer Architecture Letters* (2024). <https://doi.org/10.1109/LCA.2024.3363492>
- [17] Jianmei Dai, Zhilong Zhang, Shiwen Mao, and Danpu Liu. 2020. A View Synthesis-Based 360° VR Caching System Over MEC-Enabled C-RAN. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* (2020). <https://doi.org/10.1109/TCSVT.2019.2946755>
- [18] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. <https://doi.org/10.1109/CVPR52688.2022.01254>
- [19] Peter Deutsch. 1996. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951, RFC Editor. <https://doi.org/10.17487/RFC1951>
- [20] Jarek Duda. 2013. Asymmetric Numeral Systems: Entropy Coding Combining Speed of Huffman Coding with Compression Rate of Arithmetic Coding. *arXiv preprint arXiv: 1311.2540* (2013). <https://arxiv.org/abs/1311.2540>
- [21] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. 2024. DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation. In The 16th International Workshop on the Algorithmic Foundations of Robotics (WAFR). Springer. <https://arxiv.org/abs/2312.00583>
- [22] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In Advances in Neural Information Processing Systems (NeurIPS). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/fd881d3b625437354d4421818f81058f-Paper-Conference.pdf
- [23] Guangchi Fang and Bing Wang. 2024. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In European Conference on Computer Vision (ECCV). Springer, Cham. <https://doi.org/10.1007/978-3-031-72751-1>
- [24] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. 2019. MeshNet: mesh neural network for 3D shape representation. In AAAI Conference on Artificial Intelligence (AAAI). AAAI Press. <https://doi.org/10.1609/aaai.v33i01.33018279>
- [25] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. <https://doi.org/10.1109/CVPR52729.2023.01201>
- [26] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. 2024. Eagles: Efficient Accelerated 3D Gaussians with Lightweight Encodings. In European Conference on Computer Vision (ECCV). Springer, Cham. https://doi.org/10.1007/978-3-031-73036-8_4
- [27] Yunchoo Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing Deep Convolutional Networks Using Vector Quantization. *arXiv preprint arXiv:1412.6115* (2014). <https://doi.org/10.48550/arXiv.1412.6115>

- [28] Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. 2020. An Overview of Ongoing Point Cloud Compression Standardization Activities: Video-based (V-PCC) and Geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing (APSIPA TSIP)* (2020). <https://doi.org/10.1017/ATSiP.2020.12>
- [29] Colin Groth, Sascha Fricke, and Marcus Magnor Susana Castillo. 2023. Wavelet-Based Fast Decoding of 360-Degree Videos. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2023). <https://doi.org/10.1109/TVCG.2023.3247080>
- [30] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector Quantized Diffusion Model for Text-to-Image Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52688.2022.01043>
- [31] Hang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, , and Yi Xu. 2023. NeuRBF: A Neural Fields Representation with Adaptive Radial Basis Functions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. <https://doi.org/10.1109/ICCV51070.2023.00386>
- [32] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. 2025. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52734.2025.02006>
- [33] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. 2025. Pup 3D-GS: Principled Uncertainty Pruning for 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52734.2025.00558>
- [34] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending For Free-viewpoint Image-based Rendering. *ACM Transactions on Graphics (ToG)* (2018). <https://doi.org/10.1145/3272127.3275084>
- [35] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-out Image-based Rendering. *ACM Transactions on Graphics (ToG)* (2016). <https://doi.org/10.1145/2980179.2982420>
- [36] Adam Houenou, Philippe Bonnifait, Veronique Cherfaoui, and Wen Yao. 2013. Vehicle trajectory prediction based on motion model and maneuver recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. <https://doi.org/10.1109/IROS.2013.6696982>
- [37] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. 2022. EfficientNeRF - Efficient Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52688.2022.01256>
- [38] He Huang, Wenjie Huang, Qi Yang, Yiling Xu, and Zhu Li. 2025. A Hierarchical Compression Technique for 3D Gaussian Splatting Compression. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. <https://doi.org/10.1109/ICASSP49660.2025.10887742>
- [39] Jiangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. 2020. TPNet: Trajectory Proposal Network for Motion Prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR42600.2020.00683>
- [40] Saqib Javed, Ahmad Jarar Khan, Corentin Dumery, Chen Zhao, and Mathieu Salzmann. 2025. Temporally Compressed 3D Gaussian Splatting for Dynamic Scenes. In *British Machine Vision Conference (BMVC)*. BMVA. <https://arxiv.org/abs/2412.05700>
- [41] Yuheng Jiang, Zhehao Shen, Penghao Wang, Zhuo Su, Yu Hong, Yingliang Zhang, Jingyi Yu, and Lan Xu. 2024. HiFi4G: High-Fidelity Human Performance Rendering via Compact Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01866>
- [42] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (ToG)* (2023). <https://doi.org/10.1145/3592433>
- [43] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-scale Scene Reconstruction. *ACM Transactions on Graphics (ToG)* (2017). <https://doi.org/10.1145/3072959.3073599>
- [44] Hanyang Kong, Xingyi Yang, and Xinchao Wang. 2025. Efficient Gaussian Splatting for Monocular Dynamic Scene Rendering via Sparse Time-Variant Attribute Modeling. In *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI press. <https://doi.org/10.1609/aaai.v39i4.32460>
- [45] Sangwoon Kwak, Joonsoo Kim, Jun Young Jeong, Won-Sik Cheong, Jihyong Oh, and Munchurl Kim. 2025. MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52734.2025.01059>
- [46] Joohyun Kwon, Hanbyel Cho, and Junmo Kim. 2025. Efficient Dynamic Scene Editing via 4D Gaussian-based Static-Dynamic Separation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52734.2025.02501>
- [47] Junoh Lee, Chang-Yeon Won, Hyunjun Jung, Inhwon Bae, and Hae-Gon Jeon. 2024. Fully Explicit Dynamic Gaussian Splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/09b47a77997b7dd7d2b26bd8ff769392-Paper-Conference.pdf
- [48] Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. 2025. Optimized Minimal 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2503.16924>
- [49] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D Gaussian Representation for Radiance Field. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.02052>
- [50] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D Gaussian Splatting for Static and Dynamic Radiance Fields. *arXiv preprint arXiv:2408.03822* (2024). <https://doi.org/10.48550/arXiv.2408.03822>
- [51] Jiahui Lei, Adam Harley Yijia Weng, Leonidas Guibas, and Kostas Daniilidis. 2025. MoSe: Dynamiz Gaussian Fusion from Casual Videos via 4D Motion Scaffolds. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. <https://doi.org/10.1109/CVPR52734.2025.00578>

- [52] Changbai Li, Haodong Zhu, Hanlin Chen, Juan Zhang, Tongfei Chen, Shuo Yang, Shuwei Shao, Wenhao Dong, and Baochang Zhang. 2025. HRGS: Hierarchical Gaussian Splatting for Memory-Efficient High-Resolution 3D Reconstruction. *arXiv preprint arXiv:2506.14229* (2025). <https://arxiv.org/abs/2506.14229>
- [53] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 2025. 3D-HGS: 3D Half-Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://arxiv.org/abs/2406.02720>
- [54] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. 2025. Streaming Radiance Fields for 3D Video Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2022/file/57c2cc952f388f6185db98f441351c96-Paper-Conference.pdf
- [55] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. 2023. NerfAcc: Efficient Sampling Accelerates NeRFs. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. <https://doi.org/10.1109/ICCV51070.2023.01699>
- [56] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. 2022. Neural 3D Video Synthesis from Multi-view Video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52688.2022.00544>
- [57] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.00813>
- [58] Lei Liu, Zhenghao Chen, Wei Jiang, Wei Wang, and Dong Xu. 2024. HEMGS: A Hybrid Entropy Model for 3D Gaussian Splatting Data Compression. *arXiv preprint arXiv:2411.18473* (2024). <https://arxiv.org/abs/2411.18473>
- [59] Lei Liu, Zhenghao Chen, and Dong Xu. 2025. 3D Gaussian Splatting Data Compression with Mixture of Priors. In *ACM International Conference on Multimedia (ACM MM)*. <https://doi.org/10.1145/3746027.3755432>
- [60] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. 2024. CompGS: Efficient 3D Scene Representation via Compressed Gaussian Splatting. In *ACM International Conference on Multimedia (ACM MM)*. ACM. <https://doi.org/10.1145/3664647>
- [61] Zhenning Liu, Yingdong Hu, Xinjiu Zhang, Rui Song, Jiawei Shao, Zehong Lin, and Jun Zhang. 2024. Dynamics-Aware Gaussian Splatting Streaming Towards Fast On-the-Fly 4D Reconstruction. *arXiv preprint arXiv:2411.14847* (2024). <https://doi.org/10.48550/arXiv.2411.14847>
- [62] Tao Lu, Mulin Yu, Lining Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2024. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01952>
- [63] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *International Conference on 3D Vision (3DV)*. IEEE. <https://doi.org/10.1109/3DV62453.2024.00044>
- [64] Zhanpeng Luo, Haoxi Ran, and Li Lu. 2025. Instant4D: 4D Gaussian Splatting in Minutes. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2510.01119>
- [65] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. 2024. Gaussian Splatting SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01708>
- [66] Donald Meagher. 1982. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Computer Graphics and Image Processing* (1982). [https://doi.org/10.1016/0146-664X\(82\)90104-6](https://doi.org/10.1016/0146-664X(82)90104-6)
- [67] Takato Mizuho, Takuji Narumi, and Hideaki Kuzuoka. 2024. Reduction of Forgetting by Contextual Variation During Encoding Using 360-Degree Video-Based Immersive Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2024). <https://doi.org/10.1109/TVCG.2024.3403885>
- [68] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. 2024. Compact 3D Scene Representation via Self-organizing Gaussian Grids. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-73013-9_2
- [69] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasipour, Koohpayegani, and Hamed Pirsiavash. 2024. CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-73411-3_19
- [70] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum (CGF)* (2021). <https://doi.org/10.1111/cgf.14340>
- [71] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. 2024. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.00985>
- [72] Seungjun Oh, Younggeun Lee, Hyejin Jeon, and Eunbyung Park. 2025. Hybrid 3D-4D Gaussian Splatting for Fast Dynamic Scene Representation. *arXiv preprint arXiv:2505.13215* (2025). <https://doi.org/10.48550/arXiv.2505.13215>
- [73] Panagiotis Papantoniakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. <https://doi.org/10.1145/3651282>
- [74] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Transactions on Graphics (ToG)* (2021). <https://doi.org/10.1145/3478513.3480487>
- [75] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. Dreamfusion: Text-to-3D using 2D Diffusion. In *International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.2209.14988>

- [76] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR46437.2021.01018>
- [77] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf
- [78] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR.2017.16>
- [79] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating Diverse High-Fidelity Images with VQ-VAE-2. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf
- [80] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. <https://doi.org/10.1109/ICCV48922.2021.01407>
- [81] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. 2025. Octree-GS: Towards Consistent Real-time Rendering with Lod-structured 3D Gaussians. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2025). <https://doi.org/10.1109/TPAMI.2025.3568201>
- [82] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. OctNet: Learning Deep 3D Representations at High Resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR.2017.701>
- [83] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Remy Gendrot, Tristan Langlois, Olivier Bureller, and Arno Schubert. 2017. Dataset and Pipeline for Multi-view Light-Field Video. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. <https://doi.org/10.1109/CVPRW.2017.221>
- [84] Francesco Di Sario, Riccardo Renzulli, Marco Grangetto, Akihiro Sugimoto, and Enzo Tartaglione. 2025. GoDe: Gaussians on Demand for Progressive Level of Detail and Scalable Compression. *arXiv preprint arXiv:2501.13558* (2025). <https://arxiv.org/abs/2501.13558>
- [85] Johannes L. Schonberger and Jan-Michael Frahm. 2016. Structure-From-Motion Revisited. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR.2016.445>
- [86] Jinseok Seo, Gerard Joung hyun Kim, and Kyo Chul Kang. 1999. Levels of Detail (LoD) Engineering of VR Objects. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*. ACM. <https://doi.org/10.1145/323663.323680>
- [87] Ken Shoemake. 1985. Animating Rotation with Quaternion Curves. In *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM. <https://doi.org/doi/10.1145/325165.325242>
- [88] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2023). <https://doi.org/10.1109/TVCG.2023.3247082>
- [89] Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52688.2022.00538>
- [90] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 2024. 3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01954>
- [91] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezehraei, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR46437.2021.01120>
- [92] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. 2022. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.48550/arXiv.2202.05263>
- [93] Zhenyu Tang, Chaoran Feng, Xinhua Cheng, Wangbo Yu, Junwu Zhang, Yuan Liu, Xiaoxiao Long, Wenping Wang, and Li Yuan. 2025. NeuralGS: Bridging Neural Fields and 3D Gaussian Splatting for Compact 3D Representations. *AAAI Conference on Artificial Intelligence (AAAI)* (2025). <https://arxiv.org/abs/2503.23162>
- [94] Zhenyu Tang, Chaoran Feng, Xinhua Cheng, Wangbo Yu, Junwu Zhang, Yuan Liu, Xiaoxiao Long, Wenping Wang, and Li Yuan. 2025. NeuralGS: Bridging Neural Fields and 3D Gaussian Splatting for Compact 3D Representations. *arXiv preprint arXiv:2503.23162* (2025). <https://arxiv.org/abs/2503.23162>
- [95] Boyuan Tian, Qizhe Gao, Siran Xianyu, Xiaotong Cui, and Minjia Zhang. 2025. Flexgaussian: Flexible and Cost-effective Training-free Compression for 3D Gaussian Splatting. In *ACM International Conference on Multimedia (ACM MM)*. <https://dl.acm.org/doi/pdf/10.1145/3746027.3754744>
- [96] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. 2023. Consistent View Synthesis with Pose-Guided Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.48550/arXiv.2303.17598>
- [97] Allen Tu, Haiyang Ying, Alex Hanson, Yonghan Lee, Tom Goldstein, and Matthias Zwicker. 2025. Speedy Deformable 3D Gaussian Splatting: Fast Rendering and Compression of Dynamic Scenes. *arXiv preprint arXiv: 2506.07917* (2025). <https://arxiv.org/abs/2506.07917>
- [98] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. 2022. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.48550/arXiv.2112.10703>

- [99] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf
- [100] Evangelos Ververas, Rolandoz Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. 2024. Sags: Structure-aware 3D Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-72655-2_13
- [101] Diwen Wan, Ruijie Lu, and Gang Zeng. 2024. Superpoint Gaussian Splatting for Real-Time High-Fidelity Dynamic Scene Reconstruction. In *International Conference on Machine Learning (ICML)*. PMLR. <https://proceedings.mlr.press/v235/wan24f.html>
- [102] Chenjunjie Wang, Shashank N Sridhara, Eduardo Pavéz, Antonio Ortega, and Cheng Chang. 2025. Adaptive Voxelization for Transform Coding of 3D Gaussian Splatting Data. In *International Conference on Image Processing (ICIP)*. IEEE. <https://doi.org/10.1109/ICIP55913.2025.11084522>
- [103] Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. 2024. End-to-end Rate-distortion Optimized 3D Gaussian Representation. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-73636-0_5
- [104] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Ruprecht, and David Novotny. 2025. VGGT: Visual Geometry Grounded Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VGGT_Visual_Geometry_Grounded_Transformer_CVPR_2025_paper.html
- [105] Tao Wang, Mengyu Li, Geduo Zeng, Cheng Meng, and Qiong Zhang. 2025. Gaussian Herding across Pens: An Optimal Transport Perspective on Global Gaussian Reduction for 3DGs. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. <https://arxiv.org/abs/2506.09534>
- [106] Taorui Wang, Zitong Yu, and Yong Xu. 2025. TC-GS: Tri-plane Based Compression for 3D Gaussian Splatting. In *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. <https://arxiv.org/abs/2503.20221>
- [107] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex Kot, and Bihan Wen. 2024. ContextGS: Compact 3D Gaussian Splatting with Anchor Level Context Model. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/5c20ca4b0b20b0bd2f1d839dc605e70f-Paper-Conference.pdf
- [108] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. 2004. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing (TIP)* (2004). <https://doi.org/10.1109/TIP.2003.819861>
- [109] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01920>
- [110] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuntao Chen, Runyi Yang, Yuxin Huang, Xiaoyu Ye, Zike Yan, Yongliang Shi, Yiyi Liao, and Hao Zhao. 2024. MARS: An Instance-Aware, Modular and Realistic Simulator for Autonomous Driving. In *International Conference on Artificial Intelligence (CICAI)*. Springer, Singapore. https://doi.org/10.1007/978-981-99-8850-1_1
- [111] Yuanbo Xiangli, Lining Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahu Lin. 2022. Bungeenerf: Progressive Neural Radiance Field for Extreme Multi-scale Scene Rendering. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-19824-3_7
- [112] Shuzhao Xie, Jiahang Liu, Weixiang Zhang, Shijia Ge, Sicheng Pan, Chen Tang, Yunpeng Bai, and Zhi Wang. 2024. SizeGS: Size-aware Compression of 3D Gaussians with Hierarchical Mixed Precision Quantization. In *ACM International Conference on Multimedia (ACM MM)*. ACM. <https://doi.org/10.1145/3746027.3755370>
- [113] Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. 2024. MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation. In *European Conference on Computer Vision (ECCV)*. Springer, Cham. https://doi.org/10.1007/978-3-031-73414-4_25
- [114] Hao Xu, Xiaolin Wu, and Xi Zhang. 2025. 3DGS Compression with Sparsity-guided Hierarchical Transform Coding. *arXiv preprint arXiv:2505.22908* (2025). <https://arxiv.org/abs/2505.22908>
- [115] Lining Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, Dahu Lin, Michael Zollhöfer, and Christian Richardt. 2023. VR-NeRF: High-Fidelity Virtualized Walkable Spaces. In *SIGGRAPH Asia Conference Proceedings*. ACM. <https://doi.org/10.1145/3610548.3618139>
- [116] Zhiwen Yan, Chen Li, and Gim Hee Lee. 2023. NeRF-DS: Neural Radiance Fields for Dynamic Specular Objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52729.2023.00801>
- [117] Jiayu Yang, Weijian Su, Songqian Zhang, Yuqi Han, Jinli Suo, and Qiang Zhang. 2025. Scale-GS: Efficient Scalable Gaussian Splatting via Redundancy-filtering Training on Streaming Content. *arXiv preprint arXiv: 2508.21444* (2025). <https://arxiv.org/abs/2508.21444>
- [118] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2024. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.01922>
- [119] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2024. Real-Time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/pdf/2310.10642.pdf>
- [120] Yuheng Yuan, QiuHong Shen, Xingyi Yang, and XinChao Wang. 2025. 1000+ FPS 4D Gaussian Splatting for Dynamic Scene Rendering. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2503.16422>

- [121] Yu Ting Zhan, Cheng Yuan Ho, JHebi Yang, Yi Hsin Chen, Jui Chiu Chiang, Yu Lun Liu, and Wen Hsiao Peng. 2025. CAT-3DGS: A Context-adaptive Triplane Approach to Rate-distortion-optimized 3DGS Compression. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2503.00357>
- [122] Chen Zhang, Ganzhangqin Yuan, and Wenbing Tao. 2023. DMNet: Delaunay Meshing Network for 3D Shape Representation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. <https://doi.org/10.1109/ICCV51070.2023.01326>
- [123] Jian Zhang, S. Payandeh, and J. Dill. 2003. Levels of Detail in Reducing Cost of Haptic Rendering: A Preliminary User Study. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator System (HAPTICS)*. IEEE. <https://doi.org/10.5555/795683.797571>
- [124] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR.2018.00068>
- [125] Xinjie Zhang, Zhenning Liu, Yifan Zhang, Xingtong Ge, Dailan He, Tongda Xu, Yan Wang, Zehong Lin, and Jun Zhang Shuicheng Yan. 2025. MEGA: Memory-Efficient 4D Gaussian Splatting for Dynamic Scenes. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. <https://arxiv.org/abs/2410.13613>
- [126] Zihan Zheng, Zhenlong Wu, Houqiang Zhong, Yuan Tian, Ning Cao, Lan Xu, Jiangchao Yao, Xiaoyun Zhang, Qiang Hu, and Wenjun Zhang. 2025. 4DGCP: Efficient Hierarchical 4D Gaussian Compression for Progressive Volumetric Video Streaming. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2509.17513>
- [127] Le Hui Tianrui Chen Min Yang Xiao Tang Feng Zhu Yuchao Dai Zhicheng Lu, Xiang Guo. 2024. 3D Geometry-aware Deformable Gaussian Splatting for Dynamic View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.00850>
- [128] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. 2024. Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. <https://doi.org/10.1109/CVPR52733.2024.02048>

**SUCCESS-GS: Survey of Compactness and Compression for
Efficient Static and Dynamic Gaussian Splatting**
Supplementary Material

A NOTATION TABLE

All notations and symbols appearing in this survey are grouped into five categories according to their usage in Tab. 3. The table provides a concise reference for readers unfamiliar with common 3DGS notation, including all frequently used variables and those mentioned at least once. Vector quantities are highlighted in boldface for clarity.

Notation	Explanation
1. Basic Elements of 3D Gaussians	
μ	Center coordinates of the 3D Gaussian
s	Spatial extent occupied by the 3D Gaussian
r	Orientation of the 3D Gaussian
h	Colors that vary with the viewing direction
d	Degree of the Spherical Harmonics
Σ	Covariance matrix of the 3D Gaussian
R	Rotation matrix of the 3D Gaussian
S	Scale matrix of the 3D Gaussian
σ	Opacity value of each 3D Gaussian
M	Bit width of the 3D Gaussian attributes
G	Individual 3D Gaussian
\mathcal{G}	Set of 3D Gaussians at a specific camera pose
C	View-dependent RGB color of the Gaussian
2. Rendering and Projection	
W	Coordinate Transformation matrix from world coordinates to camera coordinates
T	Tile size
p	Pixel in 2D space
μ'	Projected center of the 3D Gaussian in 2D screen coordinates system
Σ'	Projected 2D covariance matrix of the 3D Gaussian in screen coordinates
N	Total number of Gaussians projected onto the pixel
$\alpha(p)$	Alpha blending value at pixel p
$g(p)$	Pixel value of the projected 2D Gaussian in screen coordinates
$c(p)$	Predicted pixel color value
I_{pred}	Rendered 2D image
I_{gt}	GT image
ϕ	Transformation matrix that defines the camera's viewpoint and orientation

3. Static Scene Modeling

\mathcal{A}	Gaussian attributes
M_n	Binary mask for n-th 3D Gaussian
m_n	Learnable mask parameter for n-th 3D Gaussian
ϵ	Threshold value for masking
$sg(\cdot)$	Stop Gradient operator
$\mathbb{1}[\cdot]$	Indicator function
$Sig(\cdot)$	Sigmoid activation function
\hat{s}	Masked scale of 3D Gaussian
$\hat{\sigma}$	Masked opacity of 3D Gaussian
L_m	Masking loss
GS_j	Global Significance score for j-th Gaussian
$\gamma(\Sigma_j)$	Normalized volume of j-th Gaussian
$W_{i,p}$	Influence of i-th Gaussian at pixel p
W_i	Total influence of i-th Gaussian on entire scene
α_i	Alpha blending value of i-th Gaussian
\mathcal{T}_i	Transmittance value up to the i-th 3D Gaussian
\mathcal{N}_i^K	Set of K-nearest neighbors of i-th 3D Gaussian
T_i	Appearance feature vector of i-th 3D Gaussian
$\mathcal{L}_{\text{distill}}$	Knowledge distillation loss
C_{teacher}	Rendered pixel values from teacher model
C_{student}	Rendered pixel values from student model
$S(p)$	Sensitivity of parameter p
E_j	Total energy for j-th image
p_k	Probability of k-th codebook vector
$r_{i,k}^{(s)}$	Rate loss when k-th codeword is selected for the scale of i-th 3D Gaussian
$d_{i,k}^{(s)}$	Distortion loss when k-th codeword is selected for the scale of i-th 3D Gaussian
$\mathbf{CB}^{(s)}[k]$	k-th codeword in scale codebook
P	Point clouds from COLMAP
V	Voxelized scene
f_v	Local context features of anchor
l_v	Scaling factor of anchor
O_v	Learnable offsets of anchor
\hat{f}_v	View-dependent features
δ_{vc}	Relative distance between camera and anchor
d_{vc}	View direction vector
F_a	MLP for deriving attributes
D	MLP decoder function
\hat{q}	Latent vector of the quantized attribute
\bar{q}	Rounded latent vector

f_h	Hash feature
q_i	Quantization step size for the i-th anchor
\hat{f}_a^i	Quantized i-th anchor attribute
$\phi_{\mu_a^i, \sigma_a^i}$	Gaussian distribution for i-th quantized anchor attributes
$\Phi_{\mu_a^i, \sigma_a^i}$	Cumulative distribution function for i-th anchor attributes
MLP_c	MLP for context modeling

4. Optimization and Training

x	Arbitrary point in world coordinates
I_{gt}	Ground truth image
$L_1(\cdot)$	Pixel-wise mean absolute error
λ	Weighting factor controlling relative contribution of L_1 and SSIM loss
τ_{pos}	Threshold for view-space position gradients of a Gaussian
ϵ_σ	Opacity threshold for the Gaussian

5. Dynamic Scene Modeling

Δx	Offset for position produced by time-conditioned MLP
Δr	Offset for rotation produced by time-conditioned MLP
Δs	Offset for scale produced by time-conditioned MLP
\mathbf{x}_c	Position of canonical gaussian
F_θ	Time-conditioned MLP
E_ϕ	HexPlane encoder
\mathbf{f}	Latent feature
D_ψ	Lightweight MLP decoder network
μ_{4D}	4D position
Σ_{4D}	4D covariance matrix
\mathbf{h}_{4D}	Spherical harmonics of Fourier coefficients of 4D coordinates
\mathbf{q}	Quaternion representing orientation of 3D Gaussian over time
t	Time variable for modeling dynamic scenes

Table 3. **Notation and symbols used throughout this survey.** The table organizes mathematical notation into five categories: basic elements of 3D Gaussians, rendering and projection, static scene modeling, optimization and training, and dynamic scene modeling.