Gaussian Splatting: State-of-The-Arts and Future Trends

Birendra Kathariya, Dae Yeol Lee, Tsung-Wei Huang, Tong Shao, Peng Yin, Guan-Ming Su
Dolby Laboratories, Inc., USA
{birendra.kathariya, daeyeol.lee, tsung-wei.huang, tong.shao, pyin}@dolby.com
guanmingsu@ieee.org

***Abstract*—Novel view synthesis (NVS) from a sparse set of multi-view images has long been a challenging task in computer vision community. However, with the advent of Neural Radiance Field (NeRF) and 3D Gaussian Splatting (3DGS) technologies, a remarkable progress has been made in NVS and 3D scene representation. Especially, recent progress in 3DGS has unlocked new possibilities across various domains such as Augmented-Reality (AR)/Virtual-Reality (VR), sports, entertainment, computer graphics, and 3D modeling, sparking extensive research into its practical applications. In this article, we highlight the latest advances in 3DGS works focusing on representation, optimization, and compression. The goal of this paper is to explore state-of-the-art (SoTA) 3DGS works, offering insights into emerging trends and future directions in NVS and 3D scene representation.**

***Index Terms*—Novel View Synthesis (NVS), Neural Radiance Field (NeRF), 3D Gaussian Splatting (3DGS), 3D Representation**

I. Introduction

Reconstruction of 3D scenes and novel view synthesis (NVS) from a sparse set of images captured form multiple viewpoints have always been topics of great interest among computer vision community. Photogrammetry, for example, is a classical method to reconstruct a 3D point cloud from a set of images. However, this method is not very robust and requires a large amount of images for dense 3D reconstruction. Later, Neural Radiance Field (NeRF) [1] based methods are developed to reconstruct highly robust and photo-realistic 3D scenes. However, these methods rely on implicit neural representation to learn volumetric radiance field through deep neural networks and therefore, are very slow to train and render. Hybrid methods like InstantNGP [2] improved the training and rendering speed with multi-resolution hash-grid based explicit scene representation and neural network based scene decoding. These hybrid methods achieved excellent reconstruction quality but did not reach real-time rendering.

Recently, 3D Gaussian Splatting (3DGS) has emerged as a new paradigm of learning radiance field using a set of 3D Gaussians optimized via differentiable rendering. Despite its recent introduction, 3DGS has rapidly gained attention in computer vision community, due to its remarkable rendering speed and quality. Building upon prior 3DGS surveys [3]–[6], we review more recent trends and categorize them in a structured manner aligned with an end-to-end system design for deploying 3DGS across diverse applications. We focus on the following three key areas:

1) *Representation*: This category addresses adaptations of 3DGS representation to various dimensionalities, primitive functions, and auxiliary attributes, aimed at enhancing representation capabilities and enabling advanced manipulation of 3D scenes.
2) *Optimization*: To further exploit the capability of 3DGS model, several techniques have been proposed to optimize the model parameters. In this category, we discuss the optimization techniques to address the challenges in initialization, adaptive density control, anti-aliasing, and regularization.
3) *Compression*: Exceptional rendering quality of 3DGS comes with the cost of generation of a large number of 3D Gaussian primitives. Efficient compression is required for practical deployment. We examine those research trends through the lens of 3DGS parameter reduction, compact representation, and 2D/3D codec-based compression.

II. Fundamentals of Gaussian Splatting

3D Gaussian Splatting [7] represents a volumetric scene as an ensemble of 3D Gaussians. Each 3D Gaussian is characterized by its position $\mu \in \mathbb{R}^3$, rotation in quaternion $r \in \mathbb{R}^4$, scaling $s \in \mathbb{R}^3$, opacity $\alpha \in \mathbb{R}^1$, and color represented in Spherical Harmonic (SH) coefficients $c \in \mathbb{R}^{3\times16}$, resulting in a total of 59 attributes per Gaussian. Novel view rendering is achieved by first projecting these 3D Gaussians into 2D image space based on the camera's view transformation matrix. For each pixel on the rendering plane, the final color is computed through alpha blending as follows:

$$C = \sum_{i\in N} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{1}$$

where N denotes the set of Gaussians intersecting the ray, ordered along ray direction. Here, c_i, and α_i are the color and opacity of the i-th Gaussian, and the cumulative transmittance term accounts for contributions from earlier Gaussians in the compositing process. This rendering is computationally efficient, supporting parallel alpha blending for each ray and projecting only camera-visible Gaussians, enabling real-time rendering. During training, a differentiable renderer computes gradients to optimize the Gaussian attributes, ensuring that the representation aligns accurately with the multi-view training images.

III. Representations

In this section, we summarize ongoing advancements in Gaussian splatting representations by exploring their variations and categorizing them based on key characteristics, including dimensionality, primitive functions, and auxiliary attributes. We also highlight the motivations underlying these adaptations to assist the understanding of the latest research trends.

A. Dimensionality of Gaussian

While the standard representations of Gaussians are inherently 3D, various studies have explored alternative dimensionalities to address specific application needs. For instance, 2D Gaussian Splatting (2DGS) [8] is a seminal work that overcomes the limitations of 3D Gaussians in accurately representing surfaces. It introduces 2D-oriented planar Gaussian disks and employs a ray-splat intersection for perspective-correct splatting. These surface-aligned Gaussians are particularly useful for applications like scene editing, animation, and relighting, where precise surface representation is essential for realistic manipulation and light interaction. Building on this idea, various works have incorporated depth- and normal-based supervision inspired by 2DGS to develop flat, surface-aligned Gaussians. Gaussian Opacity Field (GOF) uses such supervision on ray-Gaussian intersection planes to extract surfaces and construct meshes. Methods in [9], [10] supervise the alignment of flat Gaussians to mesh surfaces, enabling animating and editing Gaussians through mesh control. In [11], [12], surface-aligned Gaussians are constructed to provide an accurate geometric prior, enabling illumination decomposition for relighting applications. Studies have also investigated 4D Gaussian splatting, extending the representation to dynamic 3D scenes by incorporating time as an additional dimension. In [13], they introduce a spatio-temporal separable 4D rotor, enabling flexible modeling of both static and dynamic scenes. By setting the temporal dimension to zero, the rotor simplifies to a standard 3D rotation. In Space-Time Gaussian (STG) [14], the position, opacity, and rotation of the Gaussian are extended as functions of time, modeled as a 1D Gaussian and polynomial functions to incorporate temporal variations. 4G-GS [15] utilizes a hex-plane, an extension of the tri-plane into the temporal dimension, to model dynamic scenes. In [16], the authors explicitly model spatio-temporal 4D volumes using 4D Gaussian primitives, enabling NVS at any desired time stamp. There are works even extending beyond 4D, such as [17], which introduces an N-dimensional Gaussian mixture. It is projected into 3D space based on viewing conditions, enabling the handling of complex lighting and material properties.

B. Primitive Functions

Some works address the inherent limitations of 3D Gaussian kernels in representing sharp textures and edges due to their low-pass characteristics, exploring alternative primitive functions to replace Gaussians. In GES [18], the author proposes the Generalized Exponential Function (GEF) as a replacement for Gaussian functions. The method includes a GEF shape parameter as an additional attribute, offering greater flexibility in adapting kernel shapes to accommodate various structures. 3D-HGS [19] introduces a splitting plane for 3D Gaussians, where the normal of the splitting plane and additional opacities are included as Gaussian attributes. Two half-Gaussians share the remaining attributes but have distinct opacities, enabling better representation of high-frequency details. NegGS [20] introduces the concept of negative Gaussians, inspired by the Difference of Gaussians (DoG), which effectively approximates complex structures. Tangram-Splatting [21] leverages all the aforementioned function types (Gaussian, GEF, and DoG), diversifying the function types to optimally represent the scene. These methods collectively reduce the number of particles required to accurately represent scenes with complex structures.

C. Attributes of Gaussians

Beyond the visual representation of 3D scenes, some works incorporate additional attributes into Gaussians to enable physics-based or semantic decomposition of the scene, supporting 3D scene editing tasks. A branch of work focuses on 'relightable' Gaussian Splatting by decomposing illumination into environment lighting and material properties, leveraging the bidirectional reflectance distribution function (BRDF) for physically-based rendering. 3iGS [22] employs a learnable vector-matrix (VM) decomposition to represent a continuous illumination grid, with each Gaussian containing BRDF features to simulate view-dependent specular. GaussianShader [23] embeds attributes such as diffuse color, specular tint, roughness, and normals within each Gaussian, while jointly learning differentiable environment light to allow accurate rendering on reflective surfaces. BiGS [24] incorporates view-dependent lighting attributes (direct/indirect transport and diffuse/directional scattering) without depending on explicit surface normals or reflectance functions, providing compatibility with 3DGS where normals are undefined. GS-ID [11] and GS-IR [12] embed attributes like metallicity, albedo, and roughness into surface-aligned Gaussians, modeling environmental light transport using baked ambient occlusion.

Another line of work focuses on transferring semantic features from 2D foundational models into 3D space as Gaussian attributes. Feature 3DGS [25] introduces a framework for distilling features from 2D foundational models, such as SAM [26] and CLIP-LSeg [27], for semantic segmentation and language-guided editing. In [28], identity encoding is added as a Gaussian attribute, supervised by SAM-generated 2D masks to enable instance-based grouping. LangSplat [29] learns hierarchical language features distilled from CLIP [30], enabling semantic segmentation with clear object boundaries. Feature Splatting [31] utilizes DINO-regularized CLIP features to embed semantic attributes into Gaussians, which are then integrated with the Material Point Method (MPM) to decompose specific objects and simulate physics-based dynamics of the selected object based on a language query. By embedding these additional attributes, the approaches extend Gaussian representations beyond visual fidelity, enabling interactive manipulation of 3D scenes.

IV. Optimization

In this section, we review the recent advances in the optimization techniques in 3DGS. For better understanding of future research trends, we group recent research works by their positions in the training pipeline into the following four categories: initialization, adaptive density control, anti-aliasing, and regularization, as in the following subsections.

A. Initialization

Typical 3DGS models initialize the Gaussian primitives from sparse Structure-from-Motion (SfM) points. When SfM is not available, random initialization can be applied, but it usually leads to worse rendered image quality. Therefore, RAIN-GS [32] proposes a sparse-large-variance random initialization strategy where fewer Gaussians are initialized but with larger variances calculated from the average neighbor distance. This strategy helps the model to gradually introduce new Gaussians during training and avoid high-frequency error introduced by the initial small Gaussians. Instead of sparsely and randomly initializing the Gaussian primitives, RadSplat [33] uses a trained neural radiance field to initialize the Gaussians. For each sampled ray, a Gaussian primitive is initialized at the median depth location with the color of the training image. Pretrained models have also been used to assist initialization. GaussianObject [34] proposes initialization with visual hull constructed from object masks in sparse views, which are acquired from pretrained segmentation models. Gaussians are randomly initialized within the visual hull with the colors taken from reference images to avoid floaters.

B. Adaptive Density Control

The 3DGS model adjusts the number of 3D Gaussian primitives through adaptive density control, which adds more Gaussians to represent the fine structure and texture of the 3D scene, and removes the inessential Gaussians that have little contribution to the rendered images. Additional error terms have been proposed to guide the adaptive density control. Rota Bulò et al. [35] propose an auxiliary per-pixel error (e.g., SSIM) as a measure for growing Gaussian primitives to represent high-texture areas. In addition, an opacity correction strategy was proposed to correct the color bias caused by the cloned Gaussian primitives. TrimGS [36] proposes evaluating the view contribution of each Gaussian primitive to the color of the rendered images, where the Gaussians with the lowest view contributions are removed.

Geometric information has also been applied to help the adaptive density control. GaussianPro [37] proposes propagating and filtering the rendered 2D depth and normal maps for creating new Gaussians. GeoGaussian [38] proposes cloning and splitting the Gaussians in the smooth areas (e.g., walls) of the scene in densification. FSGS [39] proposes a Gaussian Unpooling strategy based on a proximity graph that connects the nearest neighbors, where new Gaussians are created on the edges whose proximity scores exceed a threshold.

Additional randomness has also been introduced to assist the training process. 3DGS-MCMC [40] adds noise to the center location of each Gaussian primitive to avoid getting stuck in local minimum. The noise strength is adaptive to the opacity such that opaque Gaussian primitives, which are usually converged, get less perturbation. GaussianObject [34] further uses a 2D diffusion model to refine the images rendered by noise-injected Gaussians to acquire the reference images to improve the robustness.

C. Anti-aliasing

In the differentiable rasterization in 3DGS, the color of each pixel is estimated from the projected 2D Gaussians at the pixel centers. Therefore, the 2D Gaussians are enlarged by a constant to ensure that their sizes are no smaller than a pixel. However, these sampling operations can result in aliasing artifact when zooming in or out in novel views. Therefore, Analytic-Splatting [41] proposes an analytic formulation to approximate the integration of a 2D Gaussian within a 2D pixel area as the intensity response.

Explicit anti-aliasing constraints and operations on the 3D and 2D Gaussian can also be applied. Mip-Splatting [42] proposes a 3D smoothing filter to apply to each Gaussian primitive to ensure the spatial frequency does not exceed the sampling limits in all training images. Besides, they proposed a 2D Mip filter to adaptively adjust the 2D Gaussians to cover at least a single pixel in screen space. On the other hand, without changing the training process, SA-GS [43] proposes a training-free 2D scale-adaptive filter that can be directly applied to any pretrained 3DGS models during rendering.

Another approach of anti-aliasing is directly learning the Gaussian primitives that do not create aliasing. Yan et al. [44] proposes a multi-scale Gaussian splatting model to maintain Gaussians at different scales and select Gaussians based on their pixel coverage during rendering. Mipmap-GS [45] proposes a multi-scale training strategy with pseudo ground truth of upsampled and downsampled images to ensure rendering aliasing-free image at arbitrary scale.

D. Regularization

Additional regularization terms have been introduced to the loss function to improve the rendered image quality. FreGS [46] proposes the frequency loss term as the L1 loss of the amplitude and phase in Fourier space. Different frequency bands are weighted differently and progressively through the entire training iterations. Geometry and depth information has also been used for regularization. GeoGaussian [38] proposes a co-planar constraint to penalize the Gaussians whose normal direction is different from the neighbors. TrimGS [36] proposes the regularization using the L1 loss between the normal map derived from the rendered depth map and the original Gaussian normal. Additional monocular depth estimated from pretrained models can also be used for regularization. DNGaussian [47] and Chung et al. [48] calculate the L1 loss between rendered depth and monocular depth for regularization, where the monocular depth is normalized using the rendered depth to resolve the scaling issue.

V. Compression

3DGS generates a considerable number of 3D Gaussians to well represent even a decent sized scene. The consequence is a substantial burden in storage and communication bandwidth, which also hinders rendering speed in low-end devices. An enormous amount of emergent effort has been invested to efficiently reduce the storage-cost and improve its scalability to low-end hardware.

We dedicate this section to identify the commonly research trends for 3DGS compression techniques and compact representations to reduce the overall size of 3DGS and improve the rendering speed. We categorize the ongoing efforts into three categories: 3DGS Parameter Reduction, Compact 3DGS Representation and, 2D and 3D Codec-based Compression.

A. 3DGS Parameter Reduction

Almost all existing 3DGS compression works follow either one or a combination of three ways to reduce the 3DGS parameters: 1) Pruning, 2) Vector Quantization (VQ) and, 3) Flexible SH representation. The original 3DGS work is overly parameterized and produces redundant Gaussian primitives. Pruning based on appropriate metrics during or post training has shown to reduce tremendous redundancy in 3DGS. Similarly, scale and rotation attributes of Gaussians in a neighborhood bear similarity and therefore can be represented with a reduced set through codebooks or clustering, also known as VQ. In original 3DGS, view-dependent color is represented using SH coefficients. Out of 59 attributes per Gaussian, 48 are SH coefficients, making SH an essential target for compression with the most potential redundancy. A few works have replaced SH based color representation with hash-grid and MLP, while others follow VQ-based method and variable degree SH representation.

For example, [49] utilizes all three methods mentioned above to reduce the 3DGS parameters. It applies a learnable masking on opacity (appearance) and scale (volume) and eliminates Gaussians according to the binary mask. It also employs residual vector quantization (R-VQ) to learn geometric codebook to represent scale and rotation. To represent view-dependent color, this method proposes to use hash-grid feature and MLP. In [50], they utilize resolution and scale-aware redundancy metric to identify regions in space with redundant Gaussian primitives and cull those Gaussians which contribute very little to the final rendered image. This work also proposes to use SH with lowest possible degree as required by a given Gaussian primitive. If the variance of the color evaluated across all views is low or increasing the SH degree does not improve the quality evaluated across all views, only lower degree SH are used.

LightGaussian [51] first prunes the Gaussians through global significance score calculation where Gaussians that contribute the least to the final rendering across all views are discarded. Then it applies knowledge distillation to learn SH coefficients at lower degree and vector quantize them subsequently. [52] and [53] utilize k-means clustering to vector quantize the Gaussian attribute for faster and smaller Gaussian splatting. While [52] learns two compact codebook for shape (scale, rotation) and appearance (opacity, color) parameters, [53] learns four codebooks for each attribute.

B. Compact Representation

Some studies focus on the correlation between nearby Gaussians, proposing hierarchical or predictive Gaussian frameworks. Such framework allows 3DGS to reformat into a very compact representation with fewer parameters. Scaffold-GS [54] is a seminal work in this direction that introduced a sparse anchor points, from which per-view frustum neural Gaussians are generated from visible anchors. These Gaussians have their attributes dynamically adjusted based on both implicit and explicit anchor features and view directions. HAC [55] builds upon [54] by introducing binary hash-grid and an Adaptive Quantization Module for learned entropy coding of anchor attributes, enabling a more compact representation. [56] also follows representation similar to Scaffold-GS but anchor's reference feature is rate-constrained through inter-primitive prediction using learned entropy modeling. In [57], they adopt a hierarchical approach, where parent Gaussian attributes are stored using a hash grid, and child Gaussian attributes are predicted and generated during the rendering stage using an MLP and an attention-based mechanism. GaussianForest [58] employs a tree structure where leaf nodes contain explicit attributes like position and opacity, which can vary significantly among neighbors. Intermediate and root nodes store implicit features processed through an MLP for covariance and view-dependent color, which exhibit local smoothness and are shared among sibling Gaussians.

EAGLES [59], in slight contrast, defines quantized latent embedding instead of Gaussian attributes and uses a set of MLPs to decode all the required Gaussian attributes. Since quantized latent embedding has fewer parameters per Gaussian primitive, this approach reduces the size of overall 3DGS representation. In [60], the features are defined in the tri-planes (xy-, yz- and xz-planes). Through Gaussian mean projection onto these planes and bi-linear interpolation, feature are collected for each Gaussian primitive. Then Gaussian attributes are subsequently decoded using a few sets of MLPs. Since features are only decomposed into 3 grid planes rather than arrays of grids, 3DGS is represented with much fewer number of parameters.

C. 2D and 3D Codec-based Compression

Image (JPEG/JPEG2000/JPEG-XL) [61] and video (AVC/HEVC/VVC) [62] codecs have been long existed and are very mature. Similarly, 3D codecs (G-PCC/V-PCC) [63] for point cloud have also seen a tremendous developmental progress. Utilizing such codec comes with a benefit of compression efficiency and seamless hardware integration. Recently, efforts have been made to utilize these conventional codec to compress 3DGS. For instance, in V^3 [64], Gaussian attributes are first optimized on a frame-group basis using entropy loss to improve inter-frame consistency. Then, attributes are integer quantized and packed into separate 2D image planes after applying Morton sorting, which preserves the spatial proximity of neighboring 3D points in the 2D representation. The image planes are then compressed with AVC/H.264. Similarly, in SOG [65], 3DGS primitives are sorted based on opacity. Then the Gaussian attributes are integer quantized and organized into separate 2D square grids while discarding the excess Gaussians that do not fit into the grid. The 2D grids are sorted with Parallel Linear Alignment Sorting (PLAS) algorithm to preserve texture smoothness and are subsequently compressed with JEPG-XL.

3DGS is essentially points in space representing Gaussian primitives with attributes. It is very natural to extend existing point cloud codec, such as G-PCC/GeS-TM, to compress 3DGS and contributions [66] [67] in the recent MPEG-INVR activity made such efforts. Both methods first pre-process to quantize the Gaussians attributes to integer representation since G-PCC and GeS-TM both process point cloud in integer format. [66] uses GeS-TM, a variant of G-PCC for coding solid point cloud, as an anchor which has octree for geometry coding and RAHT for 3-dimensional attribute coding. With this approach, the Gaussian mean is coded with octree and rest of the attributes are grouped to be 3-dimensional and coded with RAHT. SH coefficients are converted to YUV color space. The attributes which are not multiple of 3 are padded with 0 for chroma (U and V) components. However, [67] has G-PCC as an anchor, so it treats each attribute as single channel reflectance and encode them with RAHT. [68] provides analysis of 3DGS compression utilizing GeS-TM proposed in [66] but reduces the SH coefficients from 48 components to just 18. After SH conversion to YUV color space, the higher degree coefficients corresponding to U and V components are discarded as they carry almost no information.

VI. Future Trends and Conclusion

The field of 3DGS has witnessed rapid advancements, driven by increasing interest and developments across various domains. In this article, we focus on recent progress in three key areas: representation, optimization, and compression. We highlight the research trends on various alternative representation formats which enable the functionality to better represent scene structure, material properties, dynamic scenes, physical interactions, and scene segmentation. Additionally, we review the latest optimization techniques designed to improve rendering quality and training speed. Finally, we explore emergent compression techniques aimed at reducing the size of 3DGS representation and accelerating rendering. This paper provides a concise yet comprehensive overview of SoTA techniques and future trends that are shaping the future of 3DGS.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, pp. 102:1–102:15, July 2022.

[3] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3d gaussian as a new vision era: A survey," *arXiv preprint arXiv:2402.07181*, 2024.

[4] Y. Bao, T. Ding, J. Huo, Y. Liu, Y. Li, W. Li, Y. Gao, and J. Luo, "3d gaussian splatting: Survey, technologies, challenges, and opportunities," *arXiv preprint arXiv:2407.17418*, 2024.

[5] G. Chen and W. Wang, "A survey on 3d gaussian splatting," *arXiv preprint arXiv:2401.03890*, 2024.

[6] M. T. Bagdasarian, P. Knoll, Y.-H. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3dgs. zip: A survey on 3d gaussian splatting compression methods," *arXiv preprint arXiv:2407.09510*, 2024.

[7] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering.," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[8] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conf. Papers*, pp. 1–11, 2024.

[9] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 5354–5363, 2024.

[10] J. Choi, Y. Lee, H. Lee, H. Kwon, and D. Manocha, "Meshgs: Adaptive mesh-aligned gaussian splatting for high-quality rendering," *arXiv preprint arXiv:2410.08941*, 2024.

[11] K. Du, Z. Liang, and Z. Wang, "Gs-id: Illumination decomposition on gaussian splatting via diffusion prior and parametric light source optimization," *arXiv preprint arXiv:2408.08524*, 2024.

[12] Z. Liang, Q. Zhang, Y. Feng, Y. Shan, and K. Jia, "Gs-ir: 3d gaussian splatting for inverse rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 21644–21653, 2024.

[13] Y. Duan, F. Wei, Q. Dai, Y. He, W. Chen, and B. Chen, "4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes," in *ACM SIGGRAPH 2024 Conf. Papers*, pp. 1–11, 2024.

[14] Z. Li, Z. Chen, Z. Li, and Y. Xu, "Spacetime gaussian feature splatting for real-time dynamic view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 8508–8520, 2024.

[15] J. Fang, T. Yi, X. Wang, L. Xie, X. Zhang, W. Liu, M. Nießner, and Q. Tian, "Fast dynamic radiance fields with time-aware neural voxels," in *SIGGRAPH Asia 2022 Conf. Papers*, 2022.

[16] Z. Yang, H. Yang, Z. Pan, and L. Zhang, "Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting," in *Proc. Int. Conf. Learn. Represent.*, 2024.

[17] S. Diolatzis, T. Zirr, A. Kuznetsov, G. Kopanas, and A. Kaplanyan, "N-dimensional gaussians for fitting of high dimensional functions," in *ACM SIGGRAPH 2024 Conf. Papers*, pp. 1–11, 2024.

[18] A. Hamdi, L. Melas-Kyriazi, J. Mai, G. Qian, R. Liu, C. Vondrick, B. Ghanem, and A. Vedaldi, "Ges: Generalized exponential splatting for efficient radiance field rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 19812–19822, 2024.

[19] H. Li, J. Liu, M. Sznaier, and O. Camps, "3d-hgs: 3d half-gaussian splatting," *arXiv preprint arXiv:2406.02720*, 2024.

[20] A. Kasymov, B. Czekaj, M. Mazur, J. Tabor, and P. Spurek, "Neggs: Negative gaussian splatting," *arXiv preprint arXiv:2405.18163*, 2024.

[21] Y. Wang, N. Zhong, M. Chen, L. Wang, and Y. Guo, "Tangram-splatting: Optimizing 3d gaussian splatting through tangram-inspired shape priors," in *Proc. ACM Int. Conf. Multimedia*, pp. 3075–3083, 2024.

[22] Z. J. Tang and T.-J. Cham, "3igs: Factorised tensorial illumination for 3d gaussian splatting," *arXiv preprint arXiv:2408.03753*, 2024.

[23] Y. Jiang, J. Tu, Y. Liu, X. Gao, X. Long, W. Wang, and Y. Ma, "Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 5322–5332, 2024.

[24] Z. Liu, Y. Guo, X. Li, B. Bickel, and R. Zhang, "Bigs: Bidirectional gaussian primitives for relightable 3d gaussian splatting," *arXiv preprint arXiv:2408.13370*, 2024.

[25] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi, "Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 21676–21685, 2024.

[26] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 4015–4026, 2023.

[27] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation," *arXiv preprint arXiv:2201.03546*, 2022.

[28] M. Ye, M. Danelljan, F. Yu, and L. Ke, "Gaussian grouping: Segment and edit anything in 3d scenes," in *Proc. Eur. Conf. Comput. Vis.*, pp. 162–179, Springer, 2025.

[29] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, "Langsplat: 3d language gaussian splatting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 20051–20060, 2024.

[30] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, pp. 8748–8763, PMLR, 2021.

[31] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, "Feature splatting: Language-driven physics-based scene synthesis and editing," *arXiv preprint arXiv:2404.01223*, 2024.

[32] J. Jung, J. Han, H. An, J. Kang, S. Park, and S. Kim, "Relaxing accurate initialization constraint for 3d gaussian splatting," *arXiv preprint arXiv:2403.09413*, 2024.

[33] M. Niemeyer, F. Manhardt, M.-J. Rakotosaona, M. Oechsle, D. Duckworth, R. Gosula, K. Tateno, J. Bates, D. Kaeser, and F. Tombari, "Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps," *arXiv preprint arXiv:2403.13806*, 2024.

[34] C. Yang, S. Li, J. Fang, R. Liang, L. Xie, X. Zhang, W. Shen, and Q. Tian, "Gaussianobject: High-quality 3d object reconstruction from four views with gaussian splatting," *ACM Trans. Graph.*, vol. 43, no. 6, pp. 1–13, 2024.

[35] S. R. Bulò, L. Porzi, and P. Kontschieder, "Revising densification in gaussian splatting," *arXiv preprint arXiv:2404.06109*, 2024.

[36] L. Fan, Y. Yang, M. Li, H. Li, and Z. Zhang, "Trim 3d gaussian splatting for accurate geometry representation," *arXiv preprint arXiv:2406.07499*, 2024.

[37] K. Cheng, X. Long, K. Yang, Y. Yao, W. Yin, Y. Ma, W. Wang, and X. Chen, "Gaussianpro: 3d gaussian splatting with progressive propagation," in *Proc. Int. Conf. Mach. Learn.*, 2024.

[38] Y. Li, C. Lyu, Y. Di, G. Zhai, G. H. Lee, and F. Tombari, "Geogaussian: Geometry-aware gaussian splatting for scene rendering," in *Proc. Eur. Conf. Comput. Vis.*, pp. 441–457, Springer, 2025.

[39] Z. Zhu, Z. Fan, Y. Jiang, and Z. Wang, "Fsgs: Real-time few-shot view synthesis using gaussian splatting," in *Proc. Eur. Conf. Comput. Vis.*, pp. 145–163, Springer, 2025.

[40] S. Kheradmand, D. Rebain, G. Sharma, W. Sun, J. Tseng, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi, "3d gaussian splatting as markov chain monte carlo," *arXiv preprint arXiv:2404.09591*, 2024.

[41] Z. Liang, Q. Zhang, W. Hu, Y. Feng, L. Zhu, and K. Jia, "Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration," *arXiv preprint arXiv:2403.11056*, 2024.

[42] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 19447–19456, 2024.

[43] X. Song, J. Zheng, S. Yuan, H.-a. Gao, J. Zhao, X. He, W. Gu, and H. Zhao, "Sa-gs: Scale-adaptive gaussian splatting for training-free anti-aliasing," *arXiv preprint arXiv:2403.19615*, 2024.

[44] Z. Yan, W. F. Low, Y. Chen, and G. H. Lee, "Multi-scale 3d gaussian splatting for anti-aliased rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 20923–20931, 2024.

[45] J. Li, Y. Shi, J. Cao, B. Ni, W. Zhang, K. Zhang, and L. Van Gool, "Mipmap-gs: Let gaussians deform with scale-specific mipmap for anti-aliasing rendering," *arXiv preprint arXiv:2408.06286*, 2024.

[46] J. Zhang, F. Zhan, M. Xu, S. Lu, and E. Xing, "Fregs: 3d gaussian splatting with progressive frequency regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 21424–21433, 2024.

[47] J. Li, J. Zhang, X. Bai, J. Zheng, X. Ning, J. Zhou, and L. Gu, "Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 20775–20785, 2024.

[48] J. Chung, J. Oh, and K. M. Lee, "Depth-regularized optimization for 3d gaussian splatting in few-shot images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 811–820, 2024.

[49] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d gaussian representation for radiance field," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 21719–21728, 2024.

[50] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis, "Reducing the memory footprint of 3d gaussian splatting," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 7, May 2024.

[51] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *arXiv e-prints*, pp. arXiv–2311, 2023.

[52] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3d gaussian splatting for accelerated novel view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 10349–10358, 2024.

[53] K. Navaneet, K. P. Meibodi, S. A. Koohpayegani, and H. Pirsiavash, "Compgs: Smaller and faster gaussian splatting with vector quantization," *Proc. Eur. Conf. Comput. Vis.*, 2024.

[54] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 20654–20664, 2024.

[55] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Hac: Hash-grid assisted context for 3d gaussian splatting compression," in *Proc. Eur. Conf. Comput. Vis.*, pp. 422–438, Springer, 2025.

[56] X. Liu, X. Wu, P. Zhang, S. Wang, Z. Li, and S. Kwong, "Compgs: Efficient 3d scene representation via compressed gaussian splatting," in *Proc. 32nd ACM Int. Conf. Multimedia*, pp. 2936–2944, 2024.

[57] J. Cao, V. Goel, C. Wang, A. Kag, J. Hu, S. Korolev, C. Jiang, S. Tulyakov, and J. Ren, "Lightweight predictive 3d gaussian splats," *arXiv preprint arXiv:2406.19434*, 2024.

[58] F. Zhang, T. Zhang, L. Zhang, H. Huang, and Y. Luo, "Gaussian-forest: Hierarchical-hybrid 3d gaussian splatting for compressed scene modeling," *arXiv preprint arXiv:2406.08759*, 2024.

[59] S. Girish, K. Gupta, and A. Shrivastava, "Eagles: Efficient accelerated 3d gaussians with lightweight encodings," in *Proc. Eur. Conf. Comput. Vis.*, pp. 54–71, Springer, 2025.

[60] M. Wu and T. Tuytelaars, "Implicit gaussian splatting with efficient multi-level tri-plane representation," *arXiv preprint arXiv:2408.10041*, 2024.

[61] J. Alakuijala, S. Boukortt, T. Ebrahimi, E. Kliuchnikov, J. Sneyers, E. Upenik, L. Vandevenne, L. Versari, and J. Wassenberg, "Benchmarking jpeg xl image compression," in *Optics, Photonics and Digital Tech. for Imaging Applications VI*, vol. 11353, pp. 187–206, SPIE, 2020.

[62] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)," *Proc. IEEE*, vol. 109, no. 9, pp. 1463–1493, 2021.

[63] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Trans. Signal Process.*, vol. 9, p. e13, 2020.

[64] P. Wang, Z. Zhang, L. Wang, K. Yao, S. Xie, J. Yu, M. Wu, and L. Xu, "V^ 3: Viewing volumetric videos on mobiles via streamable 2d dynamic gaussians," *ACM Trans. Graph.*, vol. 43, no. 6, pp. 1–13, 2024.

[65] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in *Proc. Eur. Conf. Comput. Vis.*, pp. 18–34, Springer, 2025.

[66] N. S. B. C. Gustavo Sandri, Franck Thudor, "Ges-tm as anchor for 3d gaussian coding," in *ISO/IEC JTC1/SC29 WG7 (MPEG 3D Graphics Coding and Haptics) Document m69429*, November, 2024.

[67] K. N. K. K. Kyohei Unno, Diego Fujii, "Preliminary software implementation based on g-pcc 1st edition for anchor generation of 3d gaussian coding," in *ISO/IEC JTC1/SC29 WG7 (MPEG 3D Graphics Coding and Haptics) Document m70095*, November, 2024.

[68] D. G. Alexandre Zaghetto and A. Tabatabai, "Analysis of proposal m69429 on "ges-tm as anchor for 3d gaussian," in *ISO/IEC JTC1/SC29 WG7 (MPEG 3D Graphics Coding and Haptics) Document m70260*, November, 2024.