Fast and Compact 3D Gaussian Splatting with Polarized Opacity Prior

Zi-Ming Wang[1,2], Kai-Wen Duan[1], Kowei Huang[1], Akihiro Sugimoto[2], and Shang-Hong Lai[1]

[1] National Tsing Hua University, Taiwan
{ziming614, kevin77688, st114065531}@gapp.nthu.edu.tw, lai@cs.nthu.edu.tw
[2] National Institute of Informatics, Japan
sugimoto@nii.ac.jp

Abstract. 3D Gaussian Splatting (3DGS) achieves state-of-the-art rendering quality at real-time speeds but suffers from "model bloat"—a large number of redundant, low-opacity Gaussians that inflate memory usage and training costs. This inefficiency stems from the standard "densify-then-prune" paradigm, which expands the model aggressively before relying on pruning to achieve compactness. To mitigate this problem, we present an efficient training framework that builds an **intrinsically compact** representation, replacing the conventional *densify-then-prune* cycle. Our method leverages a synergistic design: an $\mathcal{L}_2$ reconstruction loss to provide **error-proportional gradients** that stabilize optimization, and a novel **Polarized Opacity Prior (POP)** to actively manage the Gaussian population. POP steers informative primitives toward full opacity and uninformative ones toward transparency, enabling natural pruning and accelerating rendering through Early Ray Termination. Experiments on three public datasets demonstrate that our approach consistently achieves accelerated 3DGS training with significantly fewer Gaussians while maintaining comparable visual reconstruction quality. These results show that the proposed framework provides a simple and effective path toward fast and inherently compact 3DGS training.

Keywords: 3D Gaussian Splatting · Polarized Opacity Prior · $\mathcal{L}_2$ Loss · Compact Representation · Efficient Training

1 Introduction

3D Gaussian Splatting (3DGS) [14] has established a new standard for high-quality, real-time novel view synthesis. However, its practical deployment is severely hindered by "model bloat" resulting from the conventional "densify-then-prune" paradigm. This strategy typically generates millions of redundant, low-opacity Gaussians, consuming excessive memory and wasting computational resources on primitives that contribute negligibly to the final scene. Unlike most existing compression techniques that function as post-processing steps, we argue that the fundamental inefficiency lies in the training process itself—specifically,

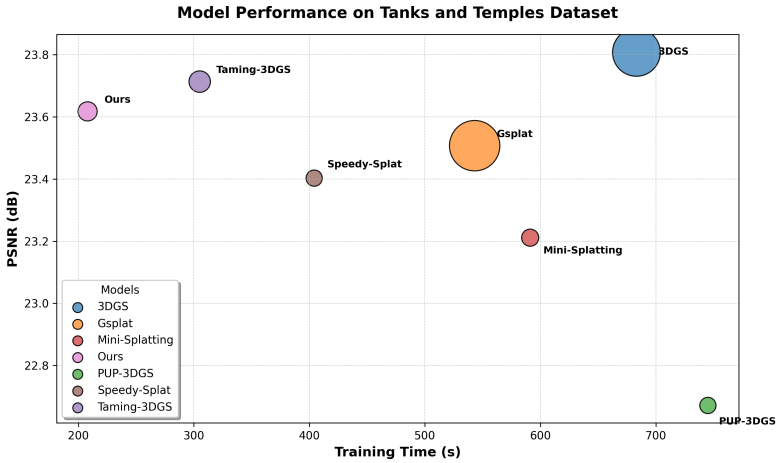


Fig. 1: Average PSNR versus training time on Tanks and Temples dataset [15]. Our method achieves the fastest training speed among state-of-the-art approaches while maintaining competitive reconstruction quality. The size of each marker indicates the number of Gaussians used in the corresponding model.

in the aggressive expansion of primitives and the reliance on heuristic "opacity resets" for pruning.

To address these challenges, we propose an efficient training framework that maintains a compact representation throughout the optimization process. Our core insight is that the stability of Gaussian evolution is tied to the relationship between reconstruction loss and opacity management. We introduce a synergistic mechanism that couples stable gradient feedback with a **Polarized Opacity Prior (POP)**, which significantly reduces Gaussian counts and accelerates convergence. The efficacy of this synergy is showcased in Fig. 1, where our approach establishes a new efficiency benchmark—attaining competitive PSNR with a drastically reduced memory footprint and shorter training times compared to current state-of-the-art methods. Specifically, our work makes the following contributions:

- **Analysis of the Gradient-Regularization Trade-off:** We identify that standard $\mathcal{L}_1$ loss provides insufficient gradient magnitude to counteract strong opacity regularization, leading to artifacts. We propose using an $\mathcal{L}_2$ formulation to provide error-proportional gradients, which stabilize optimization under sparsity constraints.
- **Polarized Opacity Prior (POP):** We introduce a novel regularization term that steers informative primitives toward full opacity while driving redundant ones toward transparency. This polarization encourages Early Ray Termination, thereby accelerating rendering.
- **Efficient Training Framework:** We develop a framework that replaces the original opacity-reset cycle with POP, enabling natural and continuous prun-

ing. This approach achieves state-of-the-art training speeds and produces significantly smaller models while delivering rendering quality comparable to current state-of-the-art methods.

2 Related Work

2.1 Novel View Synthesis and 3DGS

Neural Radiance Fields (NeRF) [20] pioneered neural implicit functions for photorealistic synthesis, yet heavy MLP evaluations limit real-time use. Subsequent research accelerated this via explicit or hybrid structures, such as voxel grids [26, 27], MLP factorization [8], and multiresolution hash encoding [22]. Notably, TensoRF [2] utilizes low-rank tensor factorization to reduce memory footprint. A major breakthrough, 3D Gaussian Splatting (3DGS) [14], employs explicit primitives and tile-based rasterization for real-time rendering. Despite its efficiency, the Gaussian count often grows explosively, leading to significant memory overhead [3, 28]. This scalability bottleneck has catalyzed research into restructuring densification to make 3DGS more compact and training-efficient.

2.2 Compact Representations for 3DGS

Existing methods generally focus on reducing redundancy through pruning and quantization, either as a post-optimization step applied after full model expansion or during joint training.

Post-processing methods utilize significance scoring or vector quantization (VQ) to compress pre-trained models. LightGaussian [6] employs a Global Significance Score based on volume and opacity, alongside knowledge distillation for spherical harmonics, to achieve over 15× compression. Compressed-3DGS [24] utilizes a sensitivity-aware vector clustering approach based on image energy contribution and linearizes Gaussians along a Morton-order space-filling curve to maximize spatial coherence, while employing quantization-aware fine-tuning alongside the clustering process, resulting in a 31× compression ratio. PUP-3DGS [11] introduces a multi-round prune-refine pipeline guided by a mathematically principled sensitivity pruning score. Derived from a second-order approximation of the reconstruction error, the score utilizes a block-wise Fisher information matrix centered on spatial parameters to quantify geometric uncertainty, thereby removing up to 90% of Gaussians with minimal quality loss.

Joint compression approaches [4, 17, 23] integrate quantization directly into the optimization loop. Methods like CompGS [23] utilize K-means-based quantization on centroids, while Compact-3DGS [17] replaces high-dimensional spherical harmonics with grid-based neural fields, incorporates residual vector quantization (R-VQ) [30] for geometric attributes, and utilizes learnable masking to dynamically prune non-essential Gaussians. Other approaches, such as Self-Organizing Gaussian Grid [21], periodically re-sort parameters into 2D grids using parallel linear assignment sorting, yielding reduction factors of 17–42×.

Finally, Mini-Splatting [7] focuses on spatial reorganization. It restructures densification through blur-based splitting and strictly constrains the primitive budget using importance-weighted sampling, thereby maintaining high-quality reconstruction with minimal primitives.

Unlike these memory-intensive *"densify-then-prune"* approaches that suffer from early Gaussian proliferation, our continuous optimization objective naturally restricts primitive growth from the outset. This ensures a stable, compact representation while maintaining comparable rendering quality.

2.3 Efficient Training of 3DGS

Another line of work focuses on reducing training latency by addressing bottlenecks ranging from low-level kernel optimizations to high-level geometric pruning. At the hardware level, DISTWAR [5] identifies that atomic operations create L2 cache contention and proposes a novel primitive for warp-level reduction, achieving a 2.44× speedup in gradient computation and 1.41× for the entire training pipeline. Moving to system-algorithm co-design, LiteGS [18] optimizes CUDA kernels using a "Cluster-Cull-Compact" pipeline that enhances spatial locality and L2 cache hit rates via Morton-code reordering, and implements warp-based rasterization. Taming-3DGS [19] targets the backpropagation bottleneck with per-splat parallelization to reduce atomic collisions alongside batched attribute updates, and replaces heuristic densification with score-based, budget-controlled sampling of Gaussian primitives. Speedy-Splat [10] integrates soft and hard pruning directly into the training loop, which is coupled with SnugBox and AccuTile for precise rendering, thereby discarding over 80% of unnecessary Gaussians to achieve a 1.47× faster training speed.

In contrast to the aforementioned methods, our framework accelerates training by processing fewer primitives. Through early suppression of uncontrolled Gaussian proliferation, we significantly reduce training overhead with only minimal modifications to low-level kernels.

2.4 Opacity Entropy Loss in 3DGS

Recent works incorporate an opacity entropy loss to binarize opacities ($\alpha \in \{0, 1\}$), driving scene primitives toward definitive physical states to precisely delineate solid 3D surface boundaries.

For instance, SuGaR [9] extracts precise polygonal meshes by aligning Gaussians with true object surfaces via SDF-based regularization. Because this formulation assumes perfectly flat and opaque primitives, an opacity entropy loss is applied to penalize intermediate values and prune semi-transparent volumes. This structural prior bypasses the sparsity issues of traditional Marching Cubes in favor of efficient depth-map sampling and Poisson reconstruction for fast and accurate meshing. Similarly, HaloGS [13] addresses geometric redundancy via a dual-representation framework that loosely couples scene geometry and appearance. It fits surface geometry with learnable triangles that decode neural

Gaussians for high-frequency appearance. To ensure compactness, HaloGS applies an opacity entropy loss during fine-tuning to binarize triangle opacities, periodically pruning primitives with low opacity or minimal view-space contribution. This structurally regularized foundation enables the efficient extraction of Level-of-Detail (LoD) planar primitives and compact meshes.

Ultimately, while both approaches demonstrate that enforcing binary opacity serves as a crucial regularizer for superior surface extraction, our work introduces a novel perspective, leveraging opacity binarization specifically to address the gradient leakage problem (Sec. 3.2) during training.

3 Methodology

3.1 Preliminary: 3D Gaussian Splatting

3DGS [14] represents a scene as a set of anisotropic Gaussians $\{\mathcal{G}_i\}$. Each Gaussian is parameterized by its position $\boldsymbol{\mu}_i$, color $\mathbf{c}_i$, opacity $o_i \in [0,1]$, and a covariance matrix $\boldsymbol{\Sigma}_i$ (factorized into a scale vector $\mathbf{s}_i$ and rotation quaternion $\mathbf{q}_i$). During rendering, these 3D Gaussians are projected onto the 2D image plane through a local affine transformation [32]. The final color of a pixel $\mathbf{p}$ is computed via alpha blending:

$$\mathbf{c}(\mathbf{p}) = \sum_{i=1}^{m} T_i \, \alpha_i \, \mathbf{c}_i, \quad \alpha_i = o_i \, \mathcal{G}_i^{2D}(\mathbf{p}). \tag{1}$$

where α_i is the per-pixel opacity of $\mathcal{G}_i$, and $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ denotes the accumulated transmittance. Here, T_i represents the remaining blending weight available for the i-th Gaussian; as more objects are encountered along the ray, T_i decreases. To optimize efficiency, rendering triggers **Early Ray Termination** if the remaining weight becomes negligible, i.e., $T_{i+1} \leq 10^{-4}$. The standard training objective minimizes a weighted combination of $\mathcal{L}_1$ and SSIM loss:

$$\mathcal{L}_{\text{color}} = (1-\lambda_{\text{SSIM}})\mathcal{L}_1 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{D-SSIM}}. \tag{2}$$

Adaptive Density Control. 3DGS is typically initialized from a sparse point cloud obtained via Structure-from-Motion (SfM) [25]. To accurately capture scene geometry, 3DGS periodically performs densification and pruning (typically every 100 iterations). Gaussians are selected for densification if their average view-space gradient magnitude exceeds a threshold τ_{Den}, as formulated in Eq. (3), where k denotes the index of the training view and $\mathbf{V}$ represents the densification period (i.e., 100). Moreover, small Gaussians undergo *cloning*, while large Gaussians are *split* into smaller primitives.

$$\frac{1}{\mathbf{V}} \sum_{k=1}^{\mathbf{V}} \sqrt{\left(\frac{\partial \mathcal{L}_k}{\partial \mu_i^{2D,x}}\right)^2 + \left(\frac{\partial \mathcal{L}_k}{\partial \mu_i^{2D,y}}\right)^2} > \tau_{\text{Den}}. \tag{3}$$

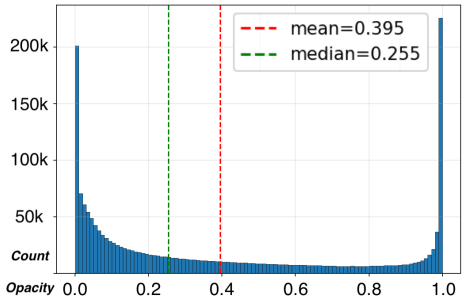


Fig. 2: Opacity statistics on the Kitchen scene [1]. Approximately 19.9% of Gaussians have opacity > 0.9, while 33.7% remain at opacity < 0.1, indicating a large number of redundant primitives.

On the other hand, redundant or poorly positioned Gaussians are *pruned* if their opacity o_i falls below a certain threshold or if they grow excessively large. A critical heuristic is the **opacity reset**—periodically setting all $o_i \to 0$ every 3,000 iterations—to force the optimization to re-justify the importance of each Gaussian, facilitating the removal of redundant primitives.

3.2 Motivation

Our approach is motivated by a critical observation: *Why do numerous redundant, low-opacity Gaussians survive the pruning process despite contributing negligibly to the final image?* (see Fig. 2).

We identify this as **Gradient Leakage**: In the standard 3DGS pipeline, even Gaussians with near-zero opacity receive persistent gradient updates accumulated from a vast number of pixels. This collective gradient often suffices to counteract the opacity reset mechanism, allowing redundant primitives to "drift" back into existence without ever providing significant geometric value.

Our hypothesis is that by driving important Gaussians toward full opacity ($o_i \to 1$), we can effectively trigger Early Ray Termination. When the accumulated transmittance T_i drops below a threshold rapidly, the rendering engine skips subsequent Gaussians in the depth order. This not only accelerates the forward pass but also halts backpropagation to redundant primitives. This creates a virtuous cycle: fewer, high-opacity Gaussians represent the scene, leading to a more compact model and faster convergence.

However, naive opacity regularization (i.e., forcing $o_i \to 1$ for all) introduces severe visual artifacts. As shown in Fig. 3 (panel b), we observe that the standard $\mathcal{L}_1 + \text{SSIM}$ loss combination may fail to correct these artifacts. This is because neither the gradient magnitude of $\mathcal{L}_1$ nor SSIM (Fig. 3, panels d and e) has a direct relationship with the reconstruction error (Fig. 3, panel c), leading to a fundamental optimization conflict with the regularization term.

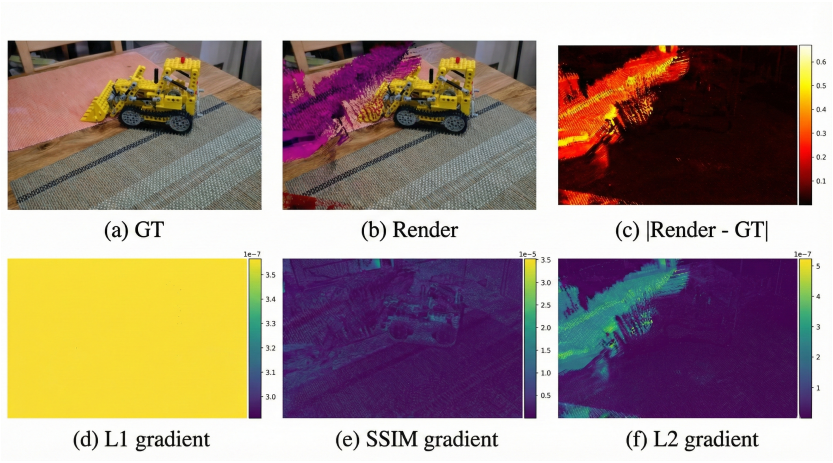


Fig. 3: Artifact demonstration on the Kitchen scene when forcing $o_i \to 1$ for all Gaussians: (a) Ground Truth, (b) Rendered image, (c) Error map $|Render - GT|$, (d) $\mathcal{L}_1$ loss gradient w.r.t. pixel color, (e) SSIM loss gradient w.r.t. pixel color, (f) $\mathcal{L}_2$ loss gradient w.r.t. pixel color.

3.3 $\mathcal{L}_2$ Loss and Blur-Based Densification

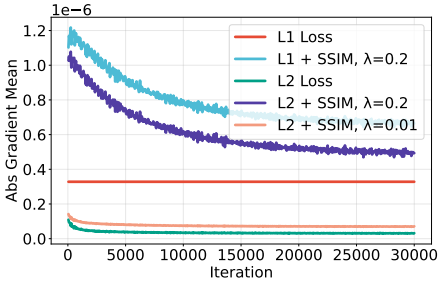
Standard 3DGS employs a combination of $\mathcal{L}_1$ and SSIM loss for scene reconstruction. However, a significant drawback of the $\mathcal{L}_1$ loss is its "error-agnostic" nature: its gradient magnitude remains constant regardless of the actual reconstruction error (Fig. 3, panel d). To address this and better suppress artifacts during optimization, we replace the primary loss with an $\mathcal{L}_2$ formulation. For a pixel $\mathbf{p}$ in an image of resolution $H \times W$, the gradients of $\mathcal{L}_1$ and $\mathcal{L}_2$ loss are:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{c}(\mathbf{p})} = \frac{\text{sign}(\Delta \mathbf{c})}{3HW}, \quad \frac{\partial \mathcal{L}_2}{\partial \mathbf{c}(\mathbf{p})} = \frac{2\Delta \mathbf{c}}{3HW}. \tag{4}$$

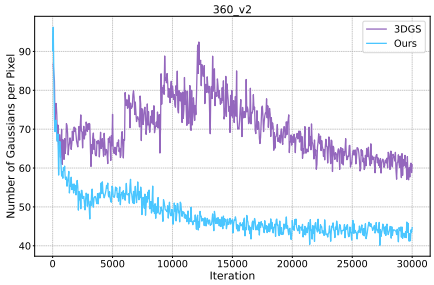
where $\Delta \mathbf{c} = \mathbf{c}(\mathbf{p}) - \mathbf{c}^{\text{GT}}(\mathbf{p})$.

As shown in Eq. (4) and Fig. 3 (panel f), the $\mathcal{L}_2$ gradient magnitude scales linearly with the color difference $\Delta \mathbf{c}$ [16]. This allows the optimization to prioritize regions with high reconstruction error, effectively stabilizing the model when driving Gaussians toward higher opacity. Depending on the error magnitude, this gradient can range from approximately $2/255$ to 2 times that of the $\mathcal{L}_1$ loss. In fact, as illustrated in Fig. 4a, the magnitude of the $\mathcal{L}_2$ gradient is typically smaller than that of the $\mathcal{L}_1$ loss under practical training conditions.

Blur-based Densification. The inherently small magnitude of $\mathcal{L}_2$ gradients poses a challenge for standard 3DGS densification, which relies on gradient-based thresholds to trigger Gaussian splitting or cloning. This limitation directly hinders the creation of new Gaussians, making the under-reconstruction problem more pronounced.



(a) Comparison of gradient magnitudes $\left|\frac{\partial \mathcal{L}}{\partial \mathbf{c}}\right|$ on the Bicycle scene



(b) Average Gaussians per pixel across iterations

Fig. 4: Effectiveness of our proposed method on Mip-NeRF 360 [1]. (a) $\mathcal{L}_2$ gradients (green) adapt to the error magnitude, whereas $\mathcal{L}_1$ gradients (red) remain constant. (b) POP significantly reduces the average Gaussians per pixel, enabling Early Ray Termination compared to the 3DGS baseline.

To address this, we implement a non-gradient-based densification mechanism inspired by Mini-Splatting [7]. Their work observes that traditional gradient-based strategies often fail in regions with smooth color transitions (i.e., **blurry areas**). We introduce a **dominant count** $\mathcal{D}_i$ to identify whether a Gaussian exerts sufficient influence over the rendered image. Specifically, during rendering, we identify the Gaussian $\mathcal{G}_i$ that provides the maximum contribution to each pixel $\mathbf{p}$:

$$i^*(\mathbf{p}) = \arg\max_i (T_i \alpha_i). \tag{5}$$

For each training view, we accumulate $\mathcal{D}_i$ for every Gaussian by counting how many pixels it dominates. A Gaussian is then selected for densification—forming the set $\mathcal{G}_{\text{blur}}$ in Eq. (6)—if its cumulative count exceeds a resolution-dependent threshold τ_{blur}:

$$\mathcal{G}_{\text{blur}} = \left\{ \mathcal{G}_i \mid \mathcal{D}_i > \tau_{\text{blur}},\ \tau_{\text{blur}} = \theta_{\text{blur}} \cdot H \cdot W \right\}. \tag{6}$$

Since the $\mathcal{L}_2$ loss yields smaller gradients compared to the $\mathcal{L}_1$ loss used in Mini-Splatting [7], we set $\theta_{\text{blur}} = 2 \times 10^{-5}$, which is one-tenth of the threshold used in their work. This adjustment allows a larger population of Gaussians—specifically the smaller ones—to participate in the densification process. Furthermore, to avoid the excessive microscopic proliferation inherent in Mini-Splatting's split-only strategy, we adaptively choose between cloning and splitting based on scale. This prevents the accumulation of uninformative primitives, ensuring a compact representation while preserving fine geometric details.

3.4 Polarized Opacity Prior (POP)

As demonstrated in Sec. 3.2, a naive, global regularization that forces $o_i \to 1$ for all Gaussians introduces severe visual artifacts. To reconcile rendering efficiency with reconstruction quality, we propose the **Polarized Opacity Prior (POP)**. Our key insight is to selectively "polarize" the opacity distribution: driving important Gaussians toward $o_i \approx 1$ to trigger Early Ray Termination, while simultaneously suppressing uninformative ones toward $o_i \approx 0$ to mitigate gradient leakage and eliminate redundancy.

Formulation. We utilize the *dominant count* $\mathcal{D}_i$ (defined in Sec. 3.3) to identify essential primitives. A binary importance mask $M_{\text{imp},i}$ is defined for each Gaussian $\mathcal{G}_i$ in a given view:

$$M_{\text{imp},i} = \mathbb{1}[\mathcal{D}_i > 0]. \tag{7}$$

Specifically, a Gaussian is classified as "important" if it serves as the primary contributor to at least one pixel in the current view. By exclusively targeting these Gaussians, we ensure the $o_i \to 1$ objective is applied only where it is geometrically and photometrically justified. The POP loss is formulated as a two-way polarization mechanism:

$$\mathcal{L}_{\text{POP}} = \frac{1}{N_{\text{imp}}} \sum_{i=1}^{N} (1 - o_i) M_{\text{imp},i} + \frac{1}{N - N_{\text{imp}}} \sum_{i=1}^{N} o_i (1 - M_{\text{imp},i}). \tag{8}$$

where N and N_{imp} denote the total and important Gaussian counts ($N_{\text{imp}} = \sum M_{\text{imp},i}$), respectively. The final training objective is:

$$\mathcal{L} = (1 - \lambda_{\text{SSIM}})\mathcal{L}_2 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{POP}}\mathcal{L}_{\text{POP}}. \tag{9}$$

In our implementation, we set $\lambda_{\text{SSIM}} = 0.01$ to balance the gradient magnitude of structural similarity with the pixel-wise intensity difference, as illustrated by the orange line in Fig. 4a. Furthermore, we apply a relatively small weight of $\lambda_{\text{POP}} = 0.001$ for the POP loss, which is sufficient to enforce opacity binarization.

Crucially, driving the opacities of important Gaussians toward 1 more readily triggers Early Ray Termination. As shown in Fig. 4b, this mechanism enables POP to significantly reduce the number of Gaussians processed per pixel, yielding substantial gains in rendering efficiency.

Gradient Modulation and Leakage Mitigation. Beyond rendering efficiency, POP enhances the optimization of scene geometry by modulating the training signal. To understand how opacity affects densification, we analyze the view-space gradient of a Gaussian $\mathcal{G}_i$ for a pixel $\mathbf{p}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_i^{2D}} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{c}(\mathbf{p})} \cdot \frac{\partial \mathbf{c}(\mathbf{p})}{\partial \alpha_i} \right) \frac{\partial \alpha_i}{\partial \boldsymbol{\mu}_i^{2D}}. \tag{10}$$

As shown in Eq. (4) and derived in the supplementary material, the terms $\frac{\partial \mathcal{L}}{\partial \mathbf{c}(\mathbf{p})}$ and $\frac{\partial \mathbf{c}(\mathbf{p})}{\partial \alpha_i}$ are independent of the opacity o_i. However, expanding the definition of α_i from Eq. (1), we see that $\alpha_i \propto o_i$:

$$\alpha_i = o_i \exp\Big(-\tfrac{1}{2}(\mathbf{p} - \boldsymbol{\mu}_i^{2D})^T (\boldsymbol{\Sigma}_i^{2D})^{-1} (\mathbf{p} - \boldsymbol{\mu}_i^{2D})\Big). \tag{11}$$

Consequently, the first term in Eq. (8) effectively amplifies the gradient feedback for important Gaussians, making them more sensitive to reconstruction errors and facilitating more effective densification in critical areas. Conversely, the second term drives uninformative Gaussians toward zero opacity. This not only attenuates their backpropagated gradients—thereby hindering them from triggering densification (Eq. (3))—but also ensures they are naturally removed by the pruning mechanism. Through this dual mechanism of suppressing redundant densification and promoting pruning, our approach successfully mitigates gradient leakage.

In particular, POP enables continuous and aggressive pruning without relying on heuristic opacity resets (Sec. 3.1). By eliminating this mechanism, our approach maintains a smooth optimization trajectory, effectively reducing both memory footprint and training time (as detailed in Sec. 5.2).

4 Experiments

4.1 Experiment Settings

Datasets and Metrics. We conduct experiments on three real-world datasets: all nine scenes from Mip-NeRF 360 [1] (four indoor and five outdoor), two outdoor scenes (*train* and *truck*) from Tanks and Temples [15], and two indoor scenes (*drjohnson* and *playroom*) from Deep Blending [12].

We evaluate rendering quality using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [31]. In addition, we report training time and the total number of Gaussians to assess model compactness and efficiency.

Implementation Details. Following the original 3DGS training schedule, we perform densification every 100 iterations from 0.5K to 15K steps. After 15K iterations, the number of Gaussians remains fixed, and only their parameters are optimized. Since the Polarized Opacity Prior (POP) primarily facilitates more effective densification, it is applied only during the first 15K iterations. Furthermore, the *blur-based densification* module is activated between 3K and 7K iterations. Additional details and ablation studies are provided in Sec. 5.

Comparisons. We compare our method with the baseline 3DGS [14], two compression-oriented methods—PUP-3DGS [11] and Mini-Splatting [7]—and two fast-training methods—Taming-3DGS [19] and Speedy-Splat [10]. Since our implementation is built upon the gsplat [29] codebase, we also include gsplat in our comparisons. For a fair evaluation, most methods are retrained and tested on the same NVIDIA RTX 4090 GPU.

Table 1: Quantitative evaluation on **Mip-NeRF 360** and **Deep Blending**. Training time and checkpoint size (Ckpt) are reported in **seconds** and **MB**, respectively. Best, second-best, and third-best results are highlighted in red, orange, and yellow, respectively.

Dataset	Mip-NeRF 360 [1]						Deep Blending [12]					
Method	Num ↓	Time ↓	Ckpt ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Num ↓	Time ↓	Ckpt ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [14]	2,509k	1,304	595	27.250	0.811	0.226	2,484k	1,197	588	29.847	0.907	0.089
Gsplat [29]	3,168k	886	713	27.661	0.824	0.166	3,042k	789	685	29.573	0.904	0.169
PUP-3DGS [11]	312k	1,216	74	26.772	0.795	0.258	282k	1,205	67	29.471	0.903	0.103
Mini-Splatting [7]	495k	896	117	27.605	0.833	0.202	348k	834	82	29.987	0.908	0.253
Taming-3DGS [19]	346k	329	82	27.147	0.772	0.294	294k	282	69	29.886	0.905	0.270
Speedy-Splat [10]	279k	728	66	26.926	0.783	0.294	250k	667	59	29.632	0.904	0.108
Ours	268k	279	60	27.013	0.760	0.277	118k	256	27	28.626	0.869	0.266

Table 2: Quantitative evaluation on **Tanks & Temples**. $*$ indicates that the method is evaluated under a reduced compression strength for fair comparison.

Dataset	Tanks & Temples					
Method	Num ↓	Time (s) ↓	Ckpt (MB) ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [14]	1,573k	683	372	23.808	0.853	0.091
Gsplat [29]	1,840k	543	414	23.508	0.845	0.127
PUP-3DGS [11]	183k	745	43	22.672	0.804	0.161
PUP-3DGS* [11]	275k	766	65	23.049	0.818	0.222
Mini-Splatting [7]	201k	591	48	23.212	0.835	0.202
Mini-Splatting* [7]	254k	613	61	23.339	0.842	0.189
Taming-3DGS [19]	318k	305	75	23.714	0.834	0.211
Speedy-Splat [10]	182k	404	43	23.403	0.819	0.142
Ours	269k	208	60	23.618	0.811	0.181

4.2 Experimental Results

The quantitative results are shown in Tab. 1 and Tab. 2. The visual results are also shown in Fig. 5. For the compression-based baselines, PUP-3DGS and Mini-Splatting, we initially follow the default configurations from their original papers. Specifically, PUP-3DGS defaults to compressing each model to one-tenth of its original size.

On the Mip-NeRF 360 and Deep Blending datasets (Tab. 1), our method achieves the fastest training time and the lowest Gaussian count among all compared methods. Notably, on the Mip-NeRF 360 dataset, our approach surpasses both PUP-3DGS (312K) and Speedy-Splat (279K) in PSNR with only 268K Gaussians.

On the Tanks and Temples dataset (Tab. 2), our method achieves the third-best PSNR score by using only ≈270K Gaussians. Notably, we attain the fastest training speed among all compared methods, reducing training time by at least 30% relative to the next fastest state-of-the-art approach, Taming-3DGS. We note that the default settings for PUP-3DGS and Mini-Splatting result in far more aggressive compression than our method. For a fair comparison of visual quality, we also report results (marked with $*$) where we adjusted their hyperpa-

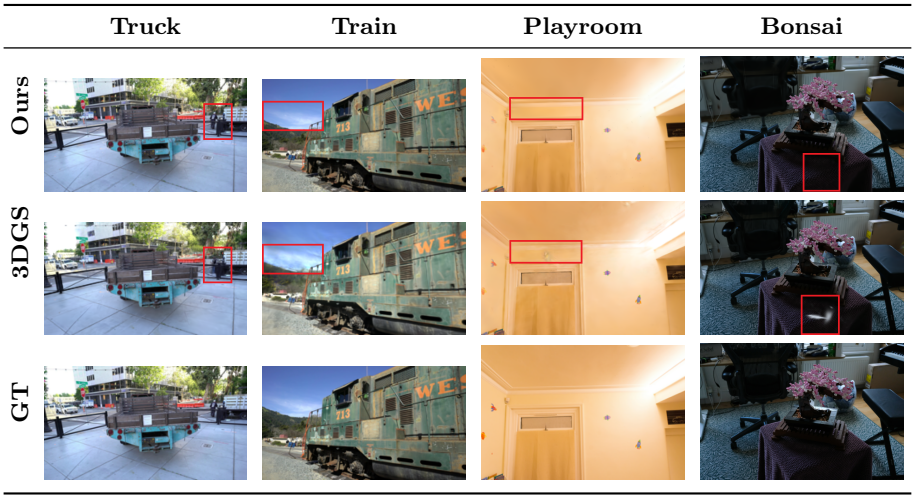


Fig. 5: Qualitative comparison on various scenes. Our method (top) achieves higher reconstruction fidelity with fewer artifacts compared to 3DGS (middle). Red boxes highlight improved fine details (e.g., Truck, Bonsai) and smoother surface reconstruction (e.g., Train, Playroom).

rameters to produce a Gaussian count comparable to ours. Specifically, we modified the PUP-3DGS schedule from 100 → 20% → 10% to 100 → 30% → 15% and increased the initial Mini-Splatting sampling factor from 0.3 to 0.4.

In summary, our method demonstrates a superior trade-off between efficiency and quality. As shown in Fig. 1, we achieve substantial reductions in training time and model size, yielding an intrinsically compact representation while maintaining comparable visual quality.

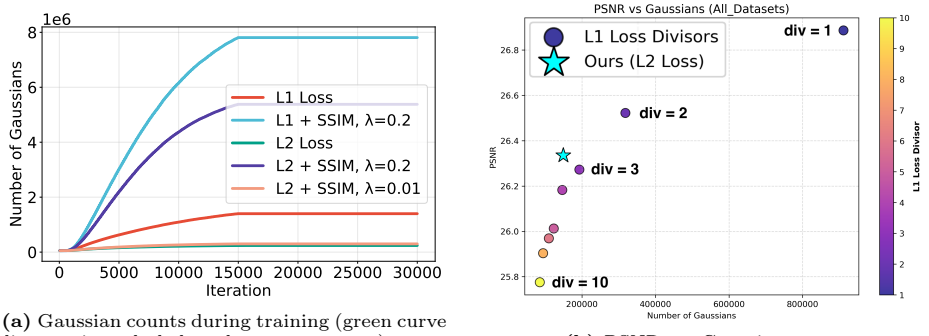
5 Ablation Study

In this section, we evaluate the effectiveness of the key components in our model: the L2 loss, blur-based densification, and Polarized Opacity Prior (POP).

5.1 $\mathcal{L}_2$ Loss and Blur-Based Densification

As discussed in Sec. 3.3, replacing the $\mathcal{L}_1$ loss with an $\mathcal{L}_2$ loss substantially reduces the overall gradient magnitude during the optimization process. This reduction directly translates to a more controlled densification process, resulting in a lower number of Gaussians, as illustrated in Fig. 6a.

Unlike the constant $\frac{\partial \mathcal{L}_1}{\partial \mathbf{c}}$ gradient, $\frac{\partial \mathcal{L}_2}{\partial \mathbf{c}}$ scales proportionally with the reconstruction error, providing more informative optimization signals. As illustrated in Fig. 6b, we compare $\mathcal{L}_2$ against $\mathcal{L}_1$ with various scaling factors (denoted as $div = n$ for $\mathcal{L}_1/n$). While simply reducing the $\mathcal{L}_1$ gradient magnitude also lowers



(a) Gaussian counts during training (green curve lies consistently below the orange curve.)

(b) PSNR vs. Gaussian count

Fig. 6: Ablation on loss functions. (a) Impact of different supervisions on Gaussian count. (b) Trade-off between PSNR and Gaussian count for $\mathcal{L}_1$ with varying scaling factors (circles) and the $\mathcal{L}_2$ loss (star).

the Gaussian count, the $\mathcal{L}_2$ loss achieves a higher PSNR at a comparable number of Gaussians. This adaptive behavior effectively suppresses artifacts and yields superior training outcomes. Given its enhanced stability and efficiency, we adopt the $\mathcal{L}_2$ + SSIM loss ($\lambda_{\text{SSIM}} = 0.01$) as our baseline for all subsequent ablations.

Blur-Based Densification. As shown in Tab. 3, relying solely on $\mathcal{L}_2$+SSIM leads to a sparse Gaussian distribution, which limits the model's representational capacity. Integrating blur-based densification effectively restores the necessary Gaussian density (Tab. 3, rows 2 and 4). This restoration process is evident in Fig. 7, where our model's Gaussian count (orange line) rapidly increases between 3k and 7k iterations.

Tab. 4 further analyzes the impact of activating blur-based densification across different training stages. We initiate this process at 3k iterations to ensure the model has reached a stable initialization. While extended schedules (e.g., 3k–15k) consume a 2–3× Gaussian budget and prolong training time, they yield inconsistent, dataset-dependent PSNR changes (+0.16 on Mip-NeRF 360, −0.07 on T&T, −0.37 on DB). On the other hand, halting too early (3k–4k) results in under-densification. Therefore, we select the **3k–7k** schedule (highlighted in **gray**), which provides the optimal trade-off: ensuring sufficient density for accurate reconstruction while avoiding unnecessary model bloat and maintaining computational efficiency.

5.2 Polarized Opacity Prior (POP)

Periodic opacity resets inevitably induce sudden performance drops and severe disruptions during training. As shown in Fig. 7, models relying on this strategy (e.g., 3DGS [14], gsplat [29], and PUP-3DGS [11]) exhibit drastic fluctuations every 3K iterations during the first 15K steps. This forces the optimization process to spend thousands of iterations recovering stability with noisy, unreliable

Table 3: Ablation on our components. "Blur" denotes our Blur-based Densification module. Our full method is highlighted in gray. **Bold** numbers indicate the best performance for each metric.

Modules			Mip-NeRF 360					Tanks & Temples					Deep Blending				
L2	Blur	POP	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓
✓	✗	✗	210k	271	26.57	0.725	0.339	221k	271	23.53	0.809	0.191	193k	259	28.51	0.870	0.252
✓	✓	✗	583k	343	**27.22**	**0.771**	**0.255**	557k	267	**23.71**	**0.821**	**0.161**	297k	275	28.56	**0.873**	**0.243**
✓	✗	✓	**140k**	**264**	26.28	0.715	0.354	**157k**	**176**	23.46	0.798	0.209	**94k**	**254**	28.43	0.867	0.272
✓	✓	✓	268k	279	27.01	0.760	0.277	269k	209	23.62	0.811	0.181	118k	257	**28.63**	0.869	0.266

Table 4: Impact of blur-based densification schedules. We highlight our balanced 3k–7k setting in gray. **Bold** numbers indicate the best performance for each metric.

Dataset	Mip-NeRF 360					Tanks & Temples					Deep Blending				
Blur Iter.	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Num ↓	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3k–4k	**175k**	**269**	26.54	0.736	0.321	**185k**	**186**	23.45	0.804	0.196	**100k**	**257**	28.21	0.863	0.271
3k–7k	268k	279	27.01	0.760	0.277	269k	209	**23.62**	0.811	0.181	118k	**257**	**28.63**	**0.869**	0.266
3k–11k	394k	326	27.13	0.768	0.260	412k	238	23.55	0.814	0.172	188k	264	28.53	**0.869**	**0.261**
3k–15k	562k	358	**27.17**	**0.773**	**0.250**	647k	275	23.55	**0.816**	**0.165**	331k	284	28.26	0.867	0.262

gradients. In contrast, POP introduces an opacity regularization term (Eq. (8)) that continuously and selectively suppresses redundant Gaussians while preserving essential ones. This mechanism eliminates such instability, maintaining a smooth training trajectory (orange curve). Furthermore, the results in Tab. 5 demonstrate that this smooth optimization trajectory allows POP to achieve comparable rendering quality using fewer Gaussians and reduced training time, thereby significantly improving overall training efficiency.

Notably, Fig. 7 shows that while Taming-3DGS [19] also exhibits gradual Gaussian growth, it relies on a complex Parabolic Schedule and Score-based Sampling for targeted densification. In contrast, our approach seamlessly synergizes with the native pruning mechanism to achieve organically controlled growth throughout training. This simplicity underscores the robustness and effectiveness of the POP design.

6 Limitations

Despite its favorable speed-compactness trade-off, our method has several limitations. First, the blur-based densification relies on a fixed schedule (3k–7k iterations), which may not generalize to scenes with diverse geometric complexities.

Second, while achieving high compactness, our model occasionally struggles with high-frequency details in complex textures. Due to the significantly reduced Gaussian count, certain artifacts or rendering gaps may become more apparent compared to the original 3DGS [14] (detailed visual comparisons are provided in the supplementary material).

Finally, despite its compactness, our model does not yet employ post-processing compression techniques—such as vector quantization or codebook-based encoding—which could further optimize storage efficiency.

Table 5: Ablation study of our **Polarized Opacity Prior (POP)** against the standard **Opacity Reset**. POP allows for a more compact representation and efficient training while maintaining competitive visual quality across all datasets.

Dataset	Mip-NeRF 360					Tanks & Temples					Deep Blending				
Method	Num↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Num↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Num↓	Time↓	PSNR↑	SSIM↑	LPIPS↓
Reset	548k	328	26.82	**0.780**	**0.260**	432k	247	22.91	0.795	0.201	234k	270	28.51	**0.871**	**0.258**
POP	**268k**	**279**	**27.01**	0.760	0.277	**269k**	**209**	**23.62**	**0.811**	**0.181**	**118k**	**257**	**28.63**	0.869	0.266

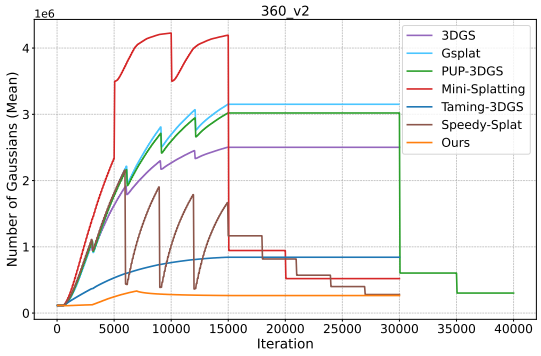


Fig. 7: Average number of Gaussians during training on the Mip-NeRF 360 [1] dataset. Note that PUP-3DGS [11] performs post-training compression after 30,000 iterations.

7 Conclusion

In this paper, we presented an efficient framework for compact 3DGS scene representation, offering a streamlined alternative to the redundant "densify-then-prune" paradigm. By integrating $\mathcal{L}_2$ loss, targeted blur-based densification, and a novel polarized opacity prior term, our method constructs a compact model from the outset—suppressing uninformative Gaussians early while selectively refining essential structures. The proposed approach significantly accelerates training and reduces model size while maintaining comparable rendering quality. Unlike techniques that rely on complex MLPs or auxiliary attributes, our design remains lightweight by adhering to the original 3DGS formulation, ensuring low architectural overhead and fast convergence. We believe this simple yet effective framework serves as a practical and robust baseline for future research in efficient 3DGS reconstruction.

Acknowledgements

This work was supported in part by the National Science and Technology Council, Taiwan, under grants NSTC 115-2634-F-007-003 and 113-2221-E-007-104-MY3.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields (2022). https://doi.org/10.1109/CVPR52688.2022.00539, https://arxiv.org/abs/2111.12077
2. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields (2022). https://doi.org/10.48550/arXiv.2203.09517, https://arxiv.org/abs/2203.09517
3. Chen, G., Wang, W.: A survey on 3d gaussian splatting (2025). https://doi.org/10.1145/3807511, https://arxiv.org/abs/2401.03890
4. Chen, Y., Wu, Q., Lin, W., Harandi, M., Cai, J.: Hac: Hash-grid assisted context for 3d gaussian splatting compression (2024). https://doi.org/10.48550/arXiv.2403.14530, https://arxiv.org/abs/2403.14530
5. Durvasula, S., Zhao, A., Chen, F., Liang, R., Sanjaya, P.K., Vijaykumar, N.: Distwar: Fast differentiable rendering on raster-based rendering pipelines (2023). https://doi.org/10.48550/arXiv.2401.05345, https://arxiv.org/abs/2401.05345
6. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps (2024). https://doi.org/10.48550/arXiv.2311.17245, https://arxiv.org/abs/2311.17245
7. Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians (2024). https://doi.org/10.1007/978-3-031-72980-5_10, https://arxiv.org/abs/2403.14166
8. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps (2021). https://doi.org/10.1109/ICCV48922.2021.01408, https://arxiv.org/abs/2103.10380
9. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering (2023), https://arxiv.org/abs/2311.12775
10. Hanson, A., Tu, A., Lin, G., Singla, V., Zwicker, M., Goldstein, T.: Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives (2025). https://doi.org/10.1109/CVPR52734.2025.00562, https://arxiv.org/abs/2412.00578
11. Hanson, A., Tu, A., Singla, V., Jayawardhana, M., Zwicker, M., Goldstein, T.: Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting (2025). https://doi.org/10.1109/CVPR52734.2025.00558, https://arxiv.org/abs/2406.10219
12. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Trans. Graph. **37**(6) (Dec 2018). https://doi.org/10.1145/3272127.3275084, https://doi.org/10.1145/3272127.3275084
13. Jiang, C., Ren, K., Xu, L., Chen, J., Pang, J., Zhang, Y., Dai, B., Yu, M.: Halogs: Loose coupling of compact geometry and gaussian splats for 3d scenes (2025), https://arxiv.org/abs/2505.20267
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering (2023). https://doi.org/10.1145/3592433, https://arxiv.org/abs/2308.04079
15. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: benchmarking large-scale scene reconstruction. ACM Trans. Graph. **36**(4) (Jul 2017). https://doi.org/10.1145/3072959.3073599, https://doi.org/10.1145/3072959.3073599

16. Lee, J.W., Lim, H., Yang, S., Choi, J.B.: Micro-splatting: Multistage isotropy-informed covariance regularization optimization for high-fidelity 3d gaussian splatting (2025). https://doi.org/10.48550/arXiv.2504.05740, https://arxiv.org/abs/2504.05740
17. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field (2024). https://doi.org/10.1109/CVPR52733.2024.02052, https://arxiv.org/abs/2311.13681
18. Liao, K., Wang, H., Chen, Z., Wang, L., Tang, Y.: Litegs: A high-performance framework to train 3dgs in subminutes via system and algorithm codesign (2025). https://doi.org/10.48550/arXiv.2503.01199, https://arxiv.org/abs/2503.01199
19. Mallick, S.S., Goel, R., Kerbl, B., Carrasco, F.V., Steinberger, M., Torre, F.D.L.: Taming 3dgs: High-quality radiance fields with limited resources (2024). https://doi.org/10.48550/arXiv.2406.15643, https://arxiv.org/abs/2406.15643
20. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis (2020). https://doi.org/10.1007/978-3-030-58452-8_24, https://arxiv.org/abs/2003.08934
21. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3D Scene Representation via Self-Organizing Gaussian Grids, p. 18–34. Springer Nature Switzerland (Nov 2024). https://doi.org/10.1007/978-3-031-73013-9_2, http://dx.doi.org/10.1007/978-3-031-73013-9_2
22. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics **41**(4), 1–15 (Jul 2022). https://doi.org/10.1145/3528223.3530127, http://dx.doi.org/10.1145/3528223.3530127
23. Navaneet, K., Meibodi, K.P., Koohpayegani, S.A., Pirsiavash, H.: Compgs: Smaller and faster gaussian splatting with vector quantization (2024). https://doi.org/10.1007/978-3-031-73411-3_19, https://arxiv.org/abs/2311.18159
24. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis (2024). https://doi.org/10.1109/CVPR52733.2024.00985, https://arxiv.org/abs/2401.02436
25. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3D. Association for Computing Machinery, New York, NY, USA, 1 edn. (2023). https://doi.org/10.1145/3596711.3596766, https://doi.org/10.1145/3596711.3596766
26. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction (2022). https://doi.org/10.48550/arXiv.2111.11215, https://arxiv.org/abs/2111.11215
27. Sun, C., Sun, M., Chen, H.T.: Improved direct voxel grid optimization for radiance fields reconstruction (2022). https://doi.org/10.48550/arXiv.2206.05085, https://arxiv.org/abs/2206.05085
28. Wu, T., Yuan, Y.J., Zhang, L.X., Yang, J., Cao, Y.P., Yan, L.Q., Gao, L.: Recent advances in 3d gaussian splatting (2024). https://doi.org/10.1007/s41095-024-0436-y, https://arxiv.org/abs/2403.11134
29. Ye, V., Li, R., Kerr, J., Turkulainen, M., Yi, B., Pan, Z., Seiskari, O., Ye, J., Hu, J., Tancik, M., Kanazawa, A.: gsplat: An open-source library for gaussian splatting (2024). https://doi.org/10.48550/arXiv.2409.06765, https://arxiv.org/abs/2409.06765

30. Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., Tagliasacchi, M.: Soundstream: An end-to-end neural audio codec (2021). https://doi.org/10.1109/taslp.2021.3129994, https://arxiv.org/abs/2107.03312
31. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric (2018). https://doi.org/10.1109/CVPR.2018.00068, https://arxiv.org/abs/1801.03924
32. Zwicker, M., Pfister, H., van Baar, J., Gatti, M.: Ewa volume splatting. In: Proceedings of the Conference on Visualization (VIS '01). pp. 83–90. IEEE Computer Society, San Diego, California, USA (2001). https://doi.org/10.1109/VIS.2001.964952, https://www.cs.umd.edu/~zwicker/publications/EWAVolumeSplatting-VIS01.pdf