Pushing Rendering Boundaries: Hard Gaussian Splatting

Qingshan Xu[1], Jiequan Cui[2*], Xuanyu Yi[1], Yuxuan Wang[1], Yuan Zhou[1], Yew-Soon Ong[1,3], Hanwang Zhang[1]

[1]Nanyang Technological University, Singapore
[2]Hefei University of Technology, China
[3]A*STAR, Singapore

Abstract

3D Gaussian Splatting (3DGS) has demonstrated impressive Novel View Synthesis (NVS) results in a real-time rendering manner. During training, it relies heavily on the average magnitude of view-space positional gradients to grow Gaussians to reduce rendering loss. However, this average operation smooths the positional gradients from different viewpoints and rendering errors from different pixels, hindering the growth and optimization of many defective Gaussians. This leads to strong spurious artifacts in some areas. To address this problem, we propose Hard Gaussian Splatting, dubbed HGS, which considers multi-view significant positional gradients and rendering errors to grow hard Gaussians that fill the gaps of classical Gaussian Splatting on 3D scenes, thus achieving superior NVS results. In detail, we present positional gradient driven HGS, which leverages multi-view significant positional gradients to uncover hard Gaussians. Moreover, we propose rendering error guided HGS, which identifies noticeable pixel rendering errors and potentially over-large Gaussians to jointly mine hard Gaussians. By growing and optimizing these hard Gaussians, our method helps to resolve blurring and needle-like artifacts. Experiments on various datasets demonstrate that our method achieves state-of-the-art rendering quality while maintaining real-time efficiency, yielding LPIPS improvements of **5.1%**, **19.7%** and **6.3%** on Mip-NeRF360, Tanks&Temples and Deep Blending, respectively.

Code — https://github.com/GhiXu/HGS

Introduction

Novel View Synthesis (NVS) is a fundamental task in computer vision and graphics, with wide applications in virtual reality, robotics and media generation. In the past few years, Neural Radiance Field (NeRF) (Mildenhall et al. 2021) has made significant advances in NVS. It adopts neural implicit representations to model 3D scenes via volume rendering (Max 1995). Generally, NeRF-based works (Barron et al. 2021, 2022; Müller et al. 2022) require ray marching for volume rendering, which compromises efficiency. Recently, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) combines a point-based explicit representation, 3D Gaussian, with GPU-friendly differentiable rasterization for volume rendering, achieving impressive real-time NVS results.

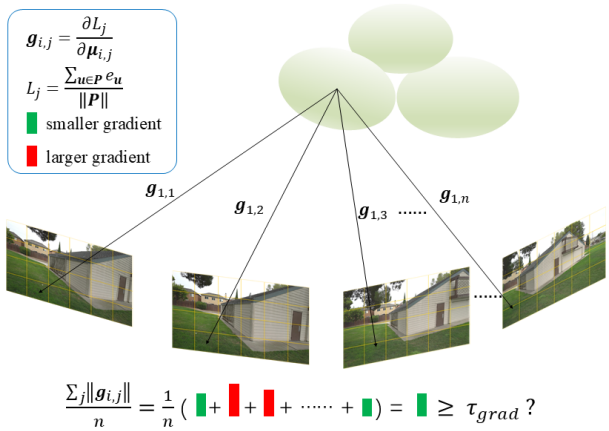
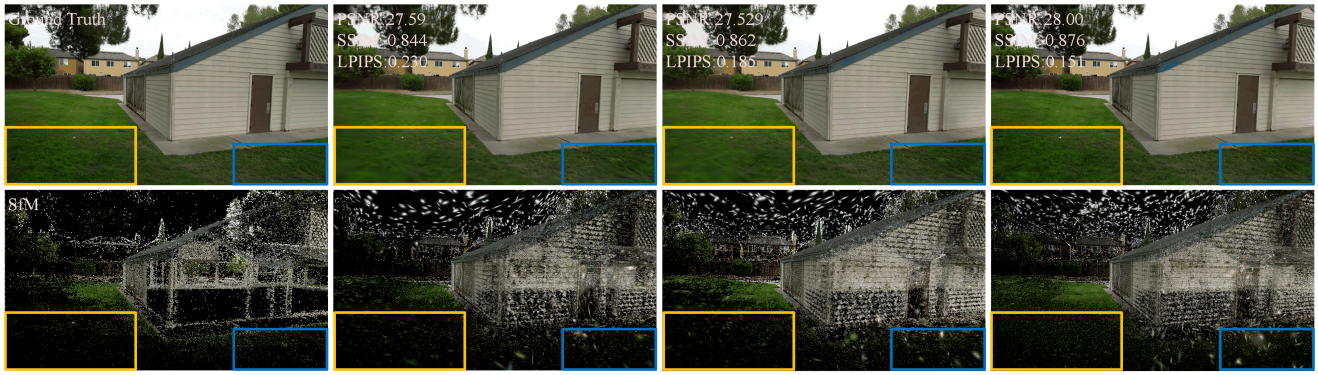


Figure 1: Illustration of the Gaussian growing criterion in 3DGS (Kerbl et al. 2023). $g_{i,j}$ denotes the positional gradient of Gaussian $\mathcal{G}_i$ under viewpoint j. The rendering loss under viewpoint j, L_j, is computed by averaging the rendering errors $\{e_u\}$ of all pixels $\boldsymbol{P}$. Larger gradients and rendering errors are smoothed by the average operation.

3DGS initializes a set of 3D Gassuains from *sparse* point clouds produced by Structure from Motion (SfM) (Snavely, Seitz, and Szeliski 2006) to represent a scene, and gradually populates empty areas by growing existing Gaussians (the bottom row of Figures 2(a) and (b)). To select suitable candidates for growing, 3DGS evaluates whether the *average* magnitude of view-space positional gradients of each Gaussian in a growth interval is larger than a threshold, as illustrated in Figure 1. This criterion encourages Gaussians to *densify* progressively and reconstruct the entire 3D scene.

However, strong spurious artifacts remain in some challenging regions, such as areas with repetitive textures and limited observations, as shown in the yellow and blue boxes in the top row of Figures 2(b) and (c). By observing the corresponding Gaussians distribution in the bottom row of Figures 2(b) and (c), we find that there are very few Gaussians in these challenging areas. This demonstrates that the growing criterion, in fact, cannot *adequately* select suitable can-

*corresponding author



(a) Reference (b) 3DGS (c) Pixel-GS (d) Ours

Figure 2: Illustration of the relationship between rendering and Gaussians distribution. We show that 3DGS and Pixel-GS (Zhang et al. 2024) fail to grow Gaussians in some challenging areas, causing artifacts in these areas. In contrast, our method effectively grows *hard* Gaussians in these challenging areas to recover 3D scenes, yielding better NVS results.

didates from existing Gaussians for growing, thus limiting the NVS performance of 3DGS. We define these candidates as *hard Gaussians*. Intuitively, the average operation used in the growing criterion prevents it from reflecting the larger positional gradients of some viewpoints, as shown in Figure 1. This leads to rendering inconsistency across views. Moreover, the rendering loss used to optimize 3DGS indicates the rendering quality of the entire image but fails to indicate the quality in local image regions. Therefore, the average magnitude of positional gradients cannot reflect noticeable rendering errors of certain local image regions, especially when these regions are only observed by few viewpoints. These deficiencies prevent 3DGS from growing *hard* Gaussians to achieve better NVS in *hard-to-learn areas*.

To address these issues, we propose Hard Gaussian Splatting, dubbed HGS, to fill the gaps of classical Gaussian Splatting on 3D scenes. Our key insight is to uncover hard Gaussians from multi-view significant positional gradients and rendering errors, and grow these Gaussians for subsequent optimization. Specifically, in addition to the original Gaussian growing criterion, we first present positional gradient driven HGS. This strategy captures k largest positional gradients for each Gaussian in the growth interval. If the minimum of these positional gradients exceeds a threshold, its corresponding Gaussian is deemed as a hard Gaussian and will grow. This strategy exploits cross-view significant positional gradients to uncover hard Gaussians, thus alleviating the rendering inconsistency across views.

In addition, we propose rendering error guided HGS. This strategy first identifies potentially over-large Gaussians that may cause blurring artifacts in the current training image. These Gaussians are then projected into the image to calculate pixel rendering errors. If noticeable pixel rendering errors are detected from multiple viewpoints within the growth interval, the corresponding Gaussian is considered as a hard Gaussian and will grow. This strategy robustly identifies noticeable pixel rendering errors across views to mine hard Gaussians, reducing rendering errors in less observed areas.

Through our proposed strategies to grow and optimize hard Gaussians, our method helps to resolve blurring and needle-like artifacts, as shown in Figure 2(d). Extensive experiments on challenging benchmark datasets (Barron et al. 2022; Knapitsch et al. 2017; Hedman et al. 2018) demonstrate that our method achieves state-of-the-art rendering quality while maintaining real-time efficiency, with LPIPS improved by **5.1%/19.7%/6.3%** on Mip-NeRF360/Tanks&Temples/Deep Blending. In summary, our main contributions are as follows:

- We analyze the deficiencies of Gaussian growing criterion–it smooths significant positional gradients and rendering errors of Gaussians. This hinders hard Gaussians growing, causing artifacts in challenging areas.
- We propose HGS, which mines hard Gaussians from multi-view significant positional gradients and rendering errors for growing and optimization. This alleviates cross-view rendering inconsistency and reduces rendering errors in less observed regions, significantly improving rendering performance.
- Our proposed HGS is general. In experiments, we show that it can boost both explicit Gaussian methods (Yu et al. 2024; Zhang et al. 2024) and neural Gaussian methods (Lu et al. 2024) to achieve better NVS results.

Related Work

Novel View Synthesis. Early image-based rendering methods (Chaurasia et al. 2013; Riegler and Koltun 2020; Kopanas et al. 2021) leverage 3D proxy geometry to generate novel views from source images. Over the past few years, NeRF (Mildenhall et al. 2021) has achieved impressive NVS results (Barron et al. 2021; Fridovich-Keil et al. 2022; Verbin et al. 2022; Yu et al. 2021; Reiser et al. 2021) and advanced other 3D tasks (Poole et al. 2022; Fu et al. 2022; Yi et al. 2024b; Sun et al. 2024; Xu et al. 2024). NeRF utilizes a multi-layer perceptron (MLP) to model scenes and leverages ray marching to render images. The expensive MLP evaluation caused by ray marching hinders real-time performance. While subsequent methods adopt more efficient representations (Müller et al. 2022; Sun, Sun, and Chen

2022; Chen et al. 2022) to accelerate the rendering process, they still struggle with real-time requirements.

Gaussian Splatting. 3DGS (Kerbl et al. 2023) employs anisotropic Gaussians (Zwicker et al. 2001) to represent 3D scenes and realizes rendering by differentiable rasterization. It stands out for its high-quality real-time rendering results and has been rapidly extended to other 3D tasks, including surface reconstruction (Yu, Sattler, and Geiger 2024; Huang et al. 2024), physical simulation (Xie et al. 2024; Zhang et al. 2025; Xu et al. 2025) and 3D generation (Tang et al. 2023; Yi et al. 2024a). To improve 3DGS, some works tackle aliasing issues by introducing 3D smoothing and 2D Mip filters (Yu et al. 2024), multi-scale Gaussians (Yan et al. 2024) and analytic integration (Liang et al. 2024). Scaffold-GS (Lu et al. 2024) builds anchor points to guide the distribution of local 3D Gaussians. Although these methods enhance 3DGS, they cannot handle strong artifacts in challenging areas.

Blurring Artifacts. Since the 3D Gaussians initialized from sparse SfM point clouds are too few to describe complete 3D scenes, 3DGS develops a growing strategy for densification and better rendering. However, large empty areas remain due to gradient collisions (Ye et al. 2024), causing blurring artifacts. To address this, some methods (Yu, Sattler, and Geiger 2024; Ye et al. 2024) accumulate the norms of individual pixel gradients or homodirectional positional gradients to better indicate regions with significant reconstruction errors. Pixel-GS (Zhang et al. 2024) weights the gradients by the number of pixels each Gaussian covers, dynamically averaging them across views to promote the growth of large Gaussians. Mini-Splatting (Fang and Wang 2024) finds that blurry artifacts are more directly related to the rendered index of Gaussians with the maximum contribution to each pixel. It then splits Gaussians with many rendered indices to reduce these artifacts. The error-based densification in (Bulò, Porzi, and Kontschieder 2024) computes rendering errors and reruns rendering to redistribute per-pixel errors to each Gaussian, increasing computation overhead. In contrast, our method mines hard Gaussians from multi-view significant positional gradients and rendering errors, and grows these Gaussians to resolve artifacts.

Preliminaries

3D Gaussian Splatting. 3DGS (Kerbl et al. 2023) represents a 3D scene with a set of anisotropic 3D Gaussians $\{\mathcal{G}_i | i = 1, \cdots, N\}$ and renders an image using α-blending with differentiable rasterization. Each Gaussian $\mathcal{G}_i$ is parameterized by a mean vector $\boldsymbol{\mu}_i \in \mathbb{R}^{3\times1}$, covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$, color $\mathbf{c}_i \in \mathbb{R}^{3\times1}$ and opacity $\alpha_i \in \mathbb{R}$ as:

$$\mathcal{G}_i(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)), \tag{1}$$

where $\mathbf{x}$ is a spatial point within the 3D scene. To render an image from viewpoint j, the 3D Gaussian $\mathcal{G}_i$ is first splatted to the 2D image plane as a 2D Gaussian $\mathcal{G}_i^{2D}$, and then α-blending is employed to compute the color of pixel $\boldsymbol{u}$ as:

$$\mathbf{c}(\boldsymbol{u}) = \sum_{i=1}^{N} w_i\mathbf{c}_i,\ w_i = \alpha_i\mathcal{G}_i^{2D}(\boldsymbol{u})\prod_{k=1}^{i-1}(1-\alpha_k\mathcal{G}_k^{2D}(\boldsymbol{u})). \tag{2}$$

Through differentiable rasterization, all the attributes in 3D Gaussians are optimized by the rendering loss. 3DGS designs an adaptive density control mechanism for Gaussian growing, adding new Gaussians where significant Gaussians are found.

Scaffold-GS. As a neural Gaussian method, Scaffold-GS (Lu et al. 2024) introduces anchor points to distribute local 3D Gaussians. Each anchor is associated with a location $\mathbf{x} \in \mathbb{R}^3$ and anchor attributes $\mathcal{A} = \{\boldsymbol{f} \in \mathbb{R}^{32}, \boldsymbol{l} \in \mathbb{R}^3, \boldsymbol{o} \in \mathbb{R}^{K\times3}\}$, where each component represents anchor context feature, scaling factor and offsets, respectively. The positions of K neural Gaussians are calculated as:

$$\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K\} = \boldsymbol{x} + \boldsymbol{o}\cdot\boldsymbol{l}. \tag{3}$$

The properties of neural Gaussians are decoded from the anchor attributes, and then used for α-blending to render images. During the Gaussian growing stage, Scaffold-GS constructs voxel grids to find significant neural Gaussians and grow their corresponding anchor points.

Deficiencies in Gaussian Growing

Both explicit and neural Gaussian methods initialize Gaussian/anchor points from sparse point clouds of SfM, which cannot completely model a 3D scene. To make explicit/neural Gaussians better represent the scene, Gaussian growing is applied to densify existing Gaussians gradually. Since both methods leverage significant Gaussians for Gaussian growing, we detail how to find significant Gaussians below.

Specifically, one Gaussian will be deemed significant when its average view-space positional gradient over M training iterations (one growth interval) satisfies:

$$\frac{\sum_j ||\boldsymbol{g}_{i,j}||}{n} \geq \tau_{\text{grad}}, \tag{4}$$

where $\boldsymbol{g}_{i,j}$ denotes the view-space positional gradient of Gaussian $\mathcal{G}_i$ under viewpoint j, n is the total number of viewpoints that Gaussian $\mathcal{G}_i$ participates in the calculation during the M iterations, and τ_{grad} is the gradient threshold. We decompose the above equation into:

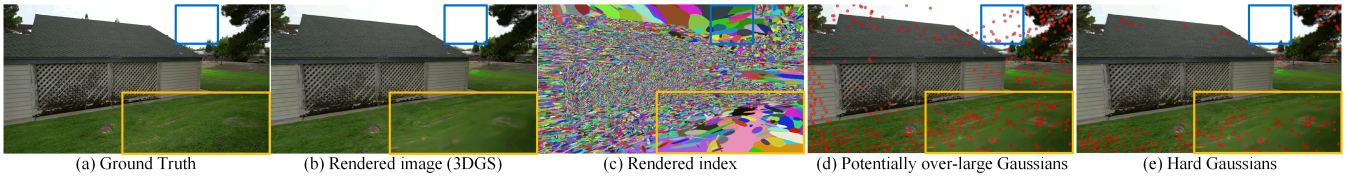
$$\frac{1}{n}(\sum_{\boldsymbol{g}_{i,j}\in\boldsymbol{G}_{\text{larger}}} ||\boldsymbol{g}_{i,j}|| + \sum_{\boldsymbol{g}_{i,j}\in\boldsymbol{G}_{\text{smaller}}} ||\boldsymbol{g}_{i,j}||) \geq \tau_{\text{grad}}, \tag{5}$$

where $\boldsymbol{G}_{\text{smaller}}$ and $\boldsymbol{G}_{\text{larger}}$ denote the set of smaller and larger gradients, respectively. In Eq. (5), $\boldsymbol{G}_{\text{smaller}}$ typically dominates, hindering some individual larger view-space positional gradients from driving Gaussians to grow, as shown in Figure 1. This makes Gaussian growing focus on most easy-to-learn viewpoints and ignore some **hard-to-learn** viewpoints, leading to cross-view rendering inconsistency.

Moreover, the above $\boldsymbol{g}_{i,j}$ is computed by:

$$\boldsymbol{g}_{i,j} = \frac{\partial L_j}{\partial \boldsymbol{\mu}_{i,j}}, \quad L_j = \frac{\sum_{\boldsymbol{u}\in\boldsymbol{P}} e_{\boldsymbol{u}}}{||\boldsymbol{P}||}, \tag{6}$$

where L_i denotes the rendering loss under viewpoint j, $\boldsymbol{\mu}_{i,j}$ is the pixel-space projection point of Gaussian $\mathcal{G}_i$ under viewpoint j, $\boldsymbol{P}$ is the total pixel coordinates, and $e_{\boldsymbol{u}}$ is the rendering error of pixel $\boldsymbol{u}$. As L_j indicates the rendering



(a) Ground Truth (b) Rendered image (3DGS) (c) Rendered index (d) Potentially over-large Gaussians (e) Hard Gaussians

Figure 3: Visual analysis of rendering error guided HGS. (a) Ground truth image. (b) Rendered image by 3DGS. (c) Rendered index of Gaussians with the maximum contribution to each pixel. (d) Projection points of potentially over-large Gaussians. (e) Projection points of hard Gaussians. Potentially over-large Gaussians may indicate false positives for Gaussian growing while hard Gaussians alleviate this (the blue boxes in (d) and (e)).

quality of the entire image instead of local image regions, the resulting average view-space positional gradient fails to reflect larger local rendering errors. This prevents Gaussians from growing in regions with larger local rendering errors. Thus, we define *Hard Gaussians* as Gaussians whose average view-space positional gradient underestimates rendering errors for specific views, causing spurious artifacts.

One straightforward way to alleviate the above problems is to lower τ_{grad} to encourage more Gaussians to grow. However, as pointed out in (Zhang et al. 2024; Ye et al. 2024), lowering τ_{grad} will prioritize densifying the well-reconstructed areas. This will introduce *redundant Gaussians* in these areas and cannot effectively reduce artifacts (see the Supplementary for more analysis). Note that, although AbsGS (Ye et al. 2024) improves the calculation of $\boldsymbol{g}_{i,j}$ (Eq. (6)) in an *absolute* way and Pixel-GS (Zhang et al. 2024) uses the number of pixels covered by Gaussians as the weight of $\boldsymbol{g}_{i,j}$ (Eq. (4)), they do not directly identify significant $e_{\boldsymbol{u}}$ or $\boldsymbol{g}_{i,j}$ in *partial* views to grow the corresponding Gaussians, limiting their rendering performance.

Hard Gaussians Splatting–HGS

To address the problems analyzed above, we propose Hard Gaussian Splatting, dubbed HGS, to uncover hard Gaussians from multi-view significant positional gradients and rendering errors. In this way, our method can grow and optimize these hard Gaussians to recover more complete 3D scenes, thereby improving rendering quality.

Positional Gradient Driven HGS

The view-space positional gradient reflects the overall image reconstruction quality under a certain viewpoint. The original Gaussian growing criterion averages the n view-space positional gradients over M iterations to find growing candidates. While averaging mitigates noise, it also smooths some individual larger positional gradients.

In fact, the larger view-space positional gradients also indicate that their corresponding Gaussians need to populate empty areas. However, they are ignored by the original growing criterion. Therefore, we present Positional Gradient driven HGS (PGHGS), which uncovers hard Gaussians from multi-view significant positional gradients. To reliably capture significant positional gradients, we first sort the n positional gradients for Gaussian $\mathcal{G}_i$ as:

$$\{\boldsymbol{g}'_{i,1}, \cdots, \boldsymbol{g}'_{i,n}\} = \text{sort}_{\downarrow}(\boldsymbol{g}_{i,1}, \cdots, \boldsymbol{g}_{i,n}), \tag{7}$$

where $\text{sort}_{\downarrow}$ denotes descending sort. Then, we mine hard Gaussians if the *k-th* largest positional gradient satisfies:

$$||\boldsymbol{g}'_{i,k}|| \geq \lambda \tau_{\text{grad}}, \tag{8}$$

where λ is a constant that controls the extent of excavation. By growing these hard Gaussians, the larger reconstruction errors under some individual viewpoints can be reduced. This facilitates the rendering consistency across views.

Rendering Error Guided HGS

By delving into the calculation of view-space positional gradient (Eq. (6)), we know that it is the gradient of the average rendering error with respect to the view-space projection point. Therefore, the view-space positional gradient cannot effectively reflect the reconstruction errors of some local image regions, especially when these regions are only observed by few viewpoints. To resolve this, we introduce Rendering Error guided HGS (REHGS) to link hard Gaussians with local rendering errors. However, as pixel rendering errors entangle contributions from multiple Gaussians (Eq. (2)), it is challenging to establish this link.

To this end, we leverage the Gaussians with the maximum contribution to correlate pixel rendering errors. Specifically, for pixel $\boldsymbol{u}$, its rendered index is defined as $\text{idx}(\boldsymbol{u}) = \arg\max_i w_i$. This considers the maximum Gaussian contribution in α-blending, thus reflecting the relationship between pixel rendering errors and Gaussians to some extent. On this basis, Gaussian $\mathcal{G}_i$ has the maximum rendering contribution on these pixels $\boldsymbol{S}_i = \{\boldsymbol{u} | \text{idx}(\boldsymbol{u}) = i, \boldsymbol{u} \in \boldsymbol{P}\}$. To detect noticeable local rendering errors rather than outlier pixels, we first identify potentially over-large Gaussians based on the size of $\boldsymbol{S}_i$. One Gaussian will be considered potentially over-large and prone to blurring artifacts if it meets the following condition (Fang and Wang 2024):

$$||\boldsymbol{S}_i|| > \tau_{\text{large}} ||\boldsymbol{P}||, \tag{9}$$

where τ_{large} is a threshold to control the extent of potentially large areas. This condition can efficiently find Gaussians that may lead to burring artifacts ("over-reconstruction"), as shown in the yellow boxes of Figures 3(c) and (d).

In fact, it is possible to describe low-textured areas and repetitive textures with potentially over-large Gaussians, as shown in the blue box of Figure 3(d). Therefore, using only Eq. (9) to determine the over-large Gaussians may result in false positives. To identify the Gaussians that cause blurring

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Method | Metric	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Plenoxels (Fridovich-Keil et al. 2022)	23.08	0.626	0.463	-	-	-	23.06	0.795	0.510
INGP (Müller et al. 2022)	25.59	0.699	0.331	-	-	-	23.62	0.797	0.423
Mip-NeRF360 (Barron et al. 2022)	27.69	0.792	0.237	-	-	-	29.40	0.901	0.245
3DGS (Kerbl et al. 2023)	27.71	0.825	0.202	24.90	0.850	0.178	29.46	0.899	0.247
Pixel-GS (Zhang et al. 2024)	27.87	0.835	0.176	25.18	0.860	0.157	28.88	0.892	0.251
Mip-Splatting (Yu et al. 2024)	27.92	**0.838**	0.175	25.02	0.857	0.166	29.25	0.903	0.240
Scaffold-GS (Lu et al. 2024)	27.98	0.825	0.208	25.74	0.861	0.170	30.23	0.907	0.245
Effi-HGS (Ours)	28.14	0.831	0.179	26.02	0.870	0.148	30.33	**0.908**	0.238
HGS (Ours)	**28.30**	0.837	**0.166**	**26.36**	**0.880**	**0.126**	**30.40**	**0.908**	**0.225**

Table 1: Quantitative comparisons on three datasets (Barron et al. 2022; Knapitsch et al. 2017; Hedman et al. 2018).

artifacts more accurately, we leverage rendering errors to mine hard Gaussians. For one potentially over-large Gaussian $\mathcal{G}_i$, we project it into the current training viewpoint j to obtain the corresponding pixel $\boldsymbol{u}_{i,j}$. This Gaussian will then be considered a hard Gaussian if it satisfies:

$$\text{SSIM}(\boldsymbol{I}_j, \hat{\boldsymbol{I}}_j)(\boldsymbol{u}_{i,j}) < \tau_{\text{SSIM}}, \tag{10}$$

where $\boldsymbol{I}_j$ and $\hat{\boldsymbol{I}}_j$ denote the ground truth and rendered images under viewpoint j, respectively. SSIM means Structural Similarity Index Measure (Wang et al. 2004), which quantifies image similarity. τ_{SSIM} is a threshold to judge rendering quality. SSIM is chosen for its local analysis capability, making it effective in detecting structural changes.

With the guidance of potentially over-large Gaussians and pixel rendering errors, our method can identify hard Gaussians in truly blurring areas, as shown in Figure 3(e). Furthermore, such Gaussians should be seen by at least two views during the M iterations. This avoids unstable growth caused by hard Gaussians determined by only a single view.

Efficient HGS

In practice, as more hard Gaussians are grown and optimized, the efficiency of Gaussian splatting will be sacrificed. To better balance rendering quality and efficiency, we introduce an efficient HGS method, called Effi-HGS. Concretely, we first count the percentage of growing Gaussians selected by the original growing criterion (Eq. (4)) in each growth interval, denoted as Q. Then, during the PGHGS, we only choose the top Q hard Gaussians for growing and optimization based on $\{\boldsymbol{g}'_{i,k} | i = 1, \cdots, N\}$. In this way, Effi-HGS can adaptively control the number of hard Gaussians for efficiency while improving rendering quality.

Experiments

Experimental Setup

Datasets and Metrics. We evaluate our methods on 17 scenes from publicly available datasets, including nine from Mip-NeRF360 (Barron et al. 2022), six from Tanks&Temples (Knapitsch et al. 2017) and two from Deep Blending (Hedman et al. 2018). The first two datasets contain both bounded indoor and unbounded outdoor scenes. The last one includes two bounded indoor scenes. We evaluate the synthesized novel views using quantitative metrics: Peak Signal-to-Noise Ratio (PSNR), SSIM and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al. 2018).

Baselines and Implementations. We compare our methods with state-of-the-art (SOTA) NeRF and Gaussian splatting methods. The NeRF methods include Plenoxels (Fridovich-Keil et al. 2022), INGP (Müller et al. 2022) and Mip-NeRF360 (Barron et al. 2022), with results from (Kerbl et al. 2023). For Gaussian splatting methods, explicit methods, including 3DGS (Kerbl et al. 2023), Mip-Splatting[1] (Yu et al. 2024) and Pixel-GS (Zhang et al. 2024), and the neural method, Scaffold-GS (Lu et al. 2024), are compared. We retrain these Gaussian splatting methods for 30K iterations on the same platform. We build our methods upon the Scaffold-GS repository (Lu et al. 2024). We maintain the same threshold τ_{grad} and growing interval M for Gaussian growing as in Scaffold-GS. We set $\{k, \lambda, \tau_{\text{large}}, \tau_{\text{SSIM}}\} = \{3, 1.0, 2\text{e-}4, 0.7\}$ across all scenes. All experiments are performed on a single NVIDIA RTX 3090 GPU.

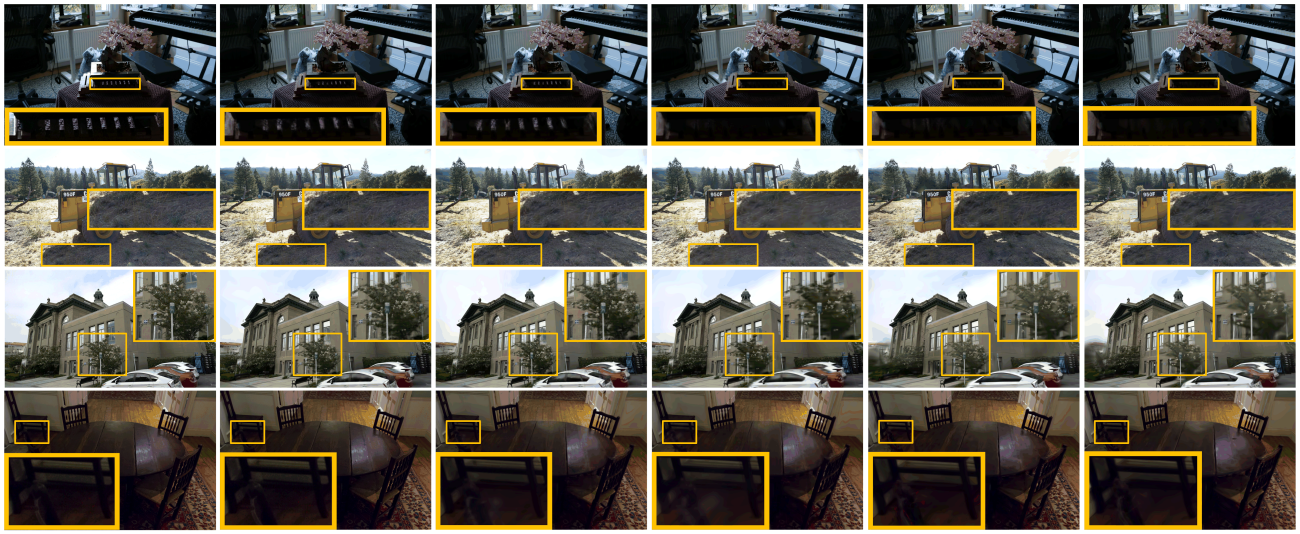
Performance Evaluation

Rendering Quality. The quantitative results are reported in Table 1. Our proposed method, HGS, achieves almost the best rendering results in all metrics on the three datasets, except for the second-best SSIM on Mip-NeRF360. Notably, in terms of LPIPS, HGS outperforms SOTA methods by **5.1%**, **19.7%**, and **6.3%** on Mip-NeRF360, Tanks&Temples and Deep Blending, respectively. Moreover, our efficient version, Effi-HGS, achieves comparable rendering results with the SOTA methods on Mip-NeRF360, and surpasses them on Tanks&Temples and Deep Blending.

The qualitative results in Figure 4 show that both our methods significantly reduce blurring and needle-like artifacts. In particular, previous methods struggle to reconstruct some occluded areas, such as those in the top row of Figure 4. In contrast, our methods successfully recover these challenging areas. This is because our methods can mine hard Gaussians from significant positional gradients and rendering errors, and grow these Gaussians to optimize fine-scale and less-observed areas.

Efficiency Comparisons. We further compare efficiency with Gaussian splatting methods in terms of training time,

[1]The Mip-Splatting we use incorporates the densification strategy from GOF. (Yu, Sattler, and Geiger 2024).



(a) Ground Truth (b) HGS (Ours) (c) Effi-HGS (Ours) (d) Scaffold-GS (e) Mip-Splatting (f) 3DGS

Figure 4: Qualitative comparisons on three datasets (Barron et al. 2022; Knapitsch et al. 2017; Hedman et al. 2018). Yellow boxes show challenging areas. Our methods can reconstruct these areas well while other methods fail.

	PSNR ↑	SSIM ↑	LPIPS ↓	Train	FPS	Storage
3DGS	24.90	0.850	0.178	**15.2m**	**159**	0.36GB
Pixel-GS	25.18	0.860	0.157	26.3m	88	0.80GB
Mip-Splatting	25.02	0.857	0.166	20.0m	126	0.45GB
Scaffold-GS	25.74	0.861	0.170	20.4m	125	**0.18GB**
Effi-HGS (Ours)	26.02	0.870	0.148	28.6m	87	0.30GB
HGS (Ours)	**26.36**	**0.880**	**0.126**	66.2m	42	0.73GB

Table 2: Efficiency comparisons on Tanks&Temples.

rendering speed and storage size in Table 2. For explicit Gaussian methods, improved rendering quality requires growing more Gaussians (resulting in larger storage size), which increases training time and reduces rendering speed. The neural Gaussian method, Scaffold-GS, can enhance both rendering quality and efficiency compared to Mip-Splatting and Pixel-GS. Similarly, since our methods will grow many hard Gaussians to improve rendering quality, this will also increase training time and reduce rendering speed. Overall, the efficiency of our methods is acceptable considering the superior rendering quality. Moreover, our efficient version achieves competitive efficiency while delivering much better rendering quality than previous methods.

Generalization Performance

To verify the generalization of our approach, several SOTA Gaussian splatting methods are integrated with our proposed HGS to evaluate NVS on Tanks&Temples. These include 3DGS, Pixel-GS, Mip-Splatting, and HAC (Chen et al. 2025), where the first three methods are explicit Gaussian methods. HAC is a compact neural method that learns structured compact hash grids for anchor attributes modeling. Considering training efficiency, we test the generalizability of our efficient HGS. The results are reported in Table 3.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS (Kerbl et al. 2023)	24.90	0.850	0.178
Pixel-GS* (Zhang et al. 2024)	25.26	0.861	0.158
Mip-Splatting* (Yu et al. 2024)	25.17	0.861	0.163
HAC (Chen et al. 2025)	25.77	0.859	0.173
3DGS + HGS	25.04 0.14 ↑	0.856 0.006 ↑	0.163 0.015 ↓
Pixel-GS* + HGS	25.28 0.02 ↑	0.864 0.003 ↑	**0.147** 0.011 ↓
Mip-Splatting* + HGS	25.26 0.09 ↑	0.862 0.001 ↑	0.155 0.008 ↓
HAC + HGS	**25.96** 0.19 ↑	**0.869** 0.010 ↑	0.150 0.023 ↓

Table 3: Generalization performance of HGS on Tanks&Temples dataset. Pixel-GS* and Mip-Splatting* mean they are combined with opacity correction.

Our proposed HGS improves all rendering metrics for both explicit and neural Gaussian methods, with notable gains in LPIPS. Note that, Pixel-GS and Mip-Splatting have grown more Gaussians than 3DGS with their modified gradients. To enhance their rendering with our HGS, we find it necessary to combine them with the opacity correction (Bulò, Porzi, and Kontschieder 2024) first to remove cloning bias, which more clearly hinders Gaussian optimization when numerous Gaussians are cloned simultaneously. With this correction, our HGS further improves rendering. Notably, the LPIPS improvements for Pixel-GS* and Mip-Splating* are noticeable, **7.0%** and **4.9%**, respectively. The results suggest that our HGS is general for both explicit and neural Gaussian methods.

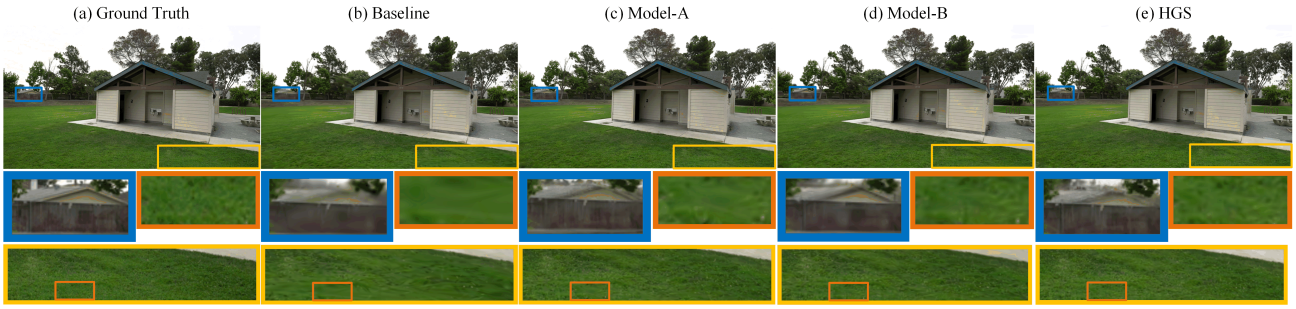


Figure 5: Qualitative results of ablation study.

	OG	PGHGS	REHGS	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline	✓			25.74	0.861	0.170
Model-A	✓	✓		26.32	0.879	0.129
Model-B	✓		✓	25.99	0.867	0.155
HGS	✓	✓	✓	**26.36**	**0.880**	**0.126**

Table 4: Ablation study on Tanks&Temples dataset. OG: Original Growing criterion.

Module	Setting	Tanks&Temples PSNR ↑	SSIM ↑	LPIPS ↓	Deep Blending PSNR ↑	SSIM ↑	LPIPS ↓
PGHGS	k=1	-	-	-	30.31	0.906	**0.211**
	k=2	26.34	**0.880**	**0.123**	30.39	**0.908**	0.219
	k=3	**26.36**	**0.880**	0.126	**30.40**	**0.908**	0.225
	k=4	26.26	0.878	0.131	30.19	**0.908**	0.230
REHGS	POG	26.26	0.879	**0.121**	30.17	0.896	**0.194**
	HG	**26.36**	**0.880**	0.126	**30.40**	**0.908**	0.225

Table 5: More module analysis on both Tanks&Temples and Deep Blending datasets. POG: Potentially Over-large Gaussians. HG: Hard Gaussians. - indicates no results due to out of memory error.

Analysis

In this section, we conduct ablation studies and analysis experiments on both Tanks&Temples and Deep Blending to evaluate the effectiveness of our proposed designs.

Ablation Study. We adopt Scaffold-GS as our baseline, progressively adding our different designs to investigate their efficacy. Results are reported in Table 4 (see supplementary for ablation study on Deep Blending). By adding PGHGS to the baseline, the rendering quality of Model-A improves significantly across all metrics (PSNR: 0.58 ↑, SSIM: 0.018 ↑, LPIPS: 0.041 ↓). As for REHGS, it clearly boosts the baseline and further enhances Model-A. This shows that both of our designs can mine hard Gaussians to improve rendering quality. Compared to PGHGS, REHGS tends to detect more noticeable rendering errors but misses some inconspicuous artifacts, which are better probed by PGHGS. Thus, the improvement brought by PGHGS is more obvious.

Figure 5 illustrates how the proposed designs improve the rendering quality. Both Model-A and Model-B can reconstruct details better than the Baseline, such as the distant house and the front grass in the blue and yellow boxes of Figure 5. Moreover, our full model further reduces blurring artifacts in Model-A, such as the orange boxes in Figures 5(c) and (e). These results show that our method effectively improves rendering performance.

Effect of k. k represents the number of significant positional gradients in the growth interval in Eq. (8). We vary this value to study its effect and report the results in Table 5. We observe that: 1) Reducing k further improves LPIPS but degrades PSNR. This is because a lower k relaxes the criterion for significant positional gradients, leading to more growing Gaussians. This may introduce noise to impair PSNR. On the other hand, more Gaussians are helpful to alleviate blurring artifacts, and LPIPS is evaluated through deep image features which have anti-noise ability. Therefore, reducing k can improve LPIPS. 2) Increasing k degrades almost all rendering metrics, suggesting that PGHGS cannot mine hard Gaussians sufficiently when k is too large.

Potentially Over-large Gaussians or Hard Gaussians. In REHGS, we identify hard Gaussians from potentially over-large Gaussians. Here, we compare the impact of these potentially over-large Gaussians (Eq. (9)) and hard Gaussians (Eq. (10)) on rendering performance, as shown in Table 4. Potentially over-large Gaussians improve LPIPS slightly but degrade PSNR a lot. As demonstrated in REHGS, potentially over-large Gaussians tend to overgrow already learned areas, making them susceptible to redundant Gaussians and, consequently, resulting in the PSNR degradation.

Conclusion

In this work, we present HGS, a hard Gaussians growing framework for Gaussian Splatting, which leverages multi-view significant positional gradients and rendering errors to uncover hard Gaussians. By growing and optimizing these hard Gaussians, our methods effectively mitigate blurring and needle-like artifacts in challenging areas. Our extensive experimental results demonstrate that our proposed HGS achieves superior rendering quality while maintaining real-time rendering speed. Moreover, we verify that our HGS can boost both explicit and neural Gaussian methods, demonstrating its potential for generalization to other Gaussian splatting related tasks.

Acknowledgments

This research is supported by National Research Foundation, Singapore, NRF-NRFI10-2024-0004, and the National Research Foundation (NRF), Singapore through the AI Singapore Programme under the project titled "AI-based Urban Cooling Technology Development" (Award No. AISG3-TC-2024-014-SGKR).

References

Barron, J. T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; and Srinivasan, P. P. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision*, 5855–5864.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5470–5479.

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

Chaurasia, G.; Duchene, S.; Sorkine-Hornung, O.; and Drettakis, G. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM transactions on graphics (TOG)*, 32(3): 1–12.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, 333–350. Springer.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2025. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, 422–438. Springer.

Fang, G.; and Wang, B. 2024. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. *arXiv preprint arXiv:2403.14166*.

Fridovich-Keil, S.; Yu, A.; Tancik, M.; Chen, Q.; Recht, B.; and Kanazawa, A. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5501–5510.

Fu, Q.; Xu, Q.; Ong, Y. S.; and Tao, W. 2022. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 35: 3403–3416.

Hedman, P.; Philip, J.; Price, T.; Frahm, J.-M.; Drettakis, G.; and Brostow, G. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6): 1–15.

Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; and Gao, S. 2024. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, 1–11.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Knapitsch, A.; Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13.

Kopanas, G.; Philip, J.; Leimkühler, T.; and Drettakis, G. 2021. Point-Based Neural Rendering with Per-View Optimization. In *Computer Graphics Forum*, volume 40, 29–43. Wiley Online Library.

Liang, Z.; Zhang, Q.; Hu, W.; Feng, Y.; Zhu, L.; and Jia, K. 2024. Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration. *arXiv preprint arXiv:2403.11056*.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Max, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2): 99–108.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15.

Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Reiser, C.; Peng, S.; Liao, Y.; and Geiger, A. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE international conference on computer vision*, 14335–14345.

Riegler, G.; and Koltun, V. 2020. Free view synthesis. In *Proceedings of the European Conference on Computer Vision*, 623–640. Springer.

Snavely, N.; Seitz, S. M.; and Szeliski, R. 2006. Photo tourism: exploring photo collections in 3D. In *ACM siggraph 2006 papers*, 835–846.

Sun, C.; Sun, M.; and Chen, H.-T. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5459–5469.

Sun, X.; Xu, Q.; Yang, X.; Zang, Y.; and Wang, C. 2024. Global and Hierarchical Geometry Consistency Priors for Few-shot NeRFs in Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 20530–20539.

Tang, J.; Ren, J.; Zhou, H.; Liu, Z.; and Zeng, G. 2023. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*.

Verbin, D.; Hedman, P.; Mildenhall, B.; Zickler, T.; Barron, J. T.; and Srinivasan, P. P. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE Conference on Computer Vision and Pattern Recognition*, 5481–5490. IEEE.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

Xie, T.; Zong, Z.; Qiu, Y.; Li, X.; Feng, Y.; Yang, Y.; and Jiang, C. 2024. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4389–4398.

Xu, Q.; Liu, J.; Yu, S.; Wang, Y.; Zhou, Y.; Zhou, J.; Cui, J.; Ong, Y.-S.; and Zhang, H. 2025. NeuSpring: Neural Spring Fields for Reconstruction and Simulation of Deformable Objects from Videos. *arXiv preprint arXiv:2511.08310*.

Xu, Q.; Yi, X.; Xu, J.; Tao, W.; Ong, Y.-S.; and Zhang, H. 2024. Few-shot NeRF by Adaptive Rendering Loss Regularization. *arXiv preprint arXiv:2410.17839*.

Yan, Z.; Low, W. F.; Chen, Y.; and Lee, G. H. 2024. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 20923–20931.

Ye, Z.; Li, W.; Liu, S.; Qiao, P.; and Dou, Y. 2024. Absgs: Recovering fine details in 3d gaussian splatting. In *ACM Multimedia 2024*.

Yi, X.; Wu, Z.; Shen, Q.; Xu, Q.; Zhou, P.; Lim, J.-H.; Yan, S.; Wang, X.; and Zhang, H. 2024a. MVGamba: Unify 3D Content Generation as State Space Sequence Modeling. *arXiv preprint arXiv:2406.06367*.

Yi, X.; Wu, Z.; Xu, Q.; Zhou, P.; Lim, J.-H.; and Zhang, H. 2024b. Diffusion time-step curriculum for one image to 3d generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9948–9958.

Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; and Kanazawa, A. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision*, 5752–5761.

Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; and Geiger, A. 2024. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 19447–19456.

Yu, Z.; Sattler, T.; and Geiger, A. 2024. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhang, T.; Yu, H.-X.; Wu, R.; Feng, B. Y.; Zheng, C.; Snavely, N.; Wu, J.; and Freeman, W. T. 2025. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, 388–406. Springer.

Zhang, Z.; Hu, W.; Lao, Y.; He, T.; and Zhao, H. 2024. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. *arXiv preprint arXiv:2403.15530*.

Zwicker, M.; Pfister, H.; Van Baar, J.; and Gross, M. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, 29–538. IEEE.