3D-HGS: 3D Half-Gaussian Splatting *

Haolin Li   Jinyang Liu   Mario Sznaier   Octavia Camps

{li.haoli, liu.jiyan, m.sznaier, o.camps}@northeastern.edu
Northeastern University
Boston, MA

Abstract

Photo-realistic image rendering from 3D scene reconstruction has advanced significantly with neural rendering techniques. Among these, 3D Gaussian Splatting (3D-GS) outperforms Neural Radiance Fields (NeRFs) in quality and speed but struggles with shape and color discontinuities. We propose 3D Half-Gaussian (3D-HGS) kernels as a plug-and-play solution to address these limitations. Our experiments show that 3D-HGS enhances existing 3D-GS methods, achieving state-of-the-art rendering quality without compromising speed. More demos and code are available at https://lihaolin88.github.io/CVPR-2025-3DHGS.

1. Introduction

The pursuit of photo-realistic and real-time rendering of 3D scenes is a core research focus in both academic and industrial sectors, with wide-ranging applications including virtual reality [12], media production [18], autonomous driving [28, 32], and extensive scene visualization [14, 16, 22]. Traditionally, meshes and point clouds have been the preferred methods for 3D scene representations due to their explicit compatibility with fast GPU/CUDA-based rasterization techniques. However, these methods often result in reconstructions of lower quality and renderings plagued by various artifacts. In contrast, recent advancements in Neural Radiance Fields (NeRF) [17] introduced continuous scene representations leveraging Multi-Layer Perceptron architectures (MLP). This approach optimizes novel-view synthesis through volumetric ray-marching techniques, providing significantly more realistic renderings. However, NeRF methods are characterized by their slow speed [5].

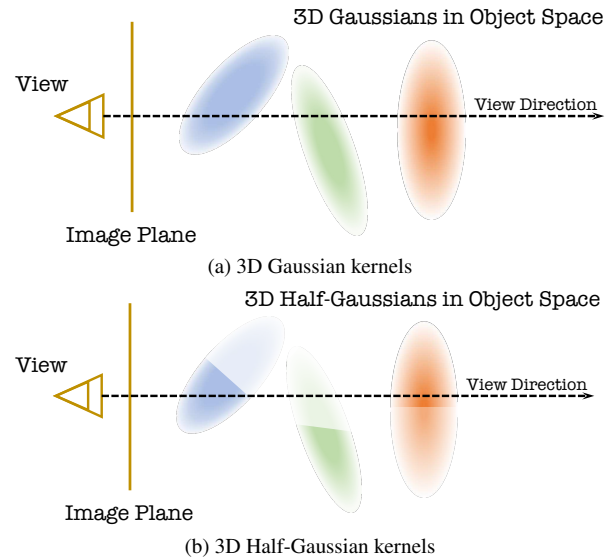(a) 3D Gaussian kernels

(b) 3D Half-Gaussian kernels

Figure 1. Illustration of the 3D-GS kernels and the proposed 3D Half-Gaussian kernels, where each half of the kernel is allowed to have different opacity parameters.

Recently, 3D Gaussian splatting (3D-GS) [9] has emerged as a state-of-the-art approach, outperforming existing methods in terms of both rendering quality and speed. The concept of using 3D Gaussians to parameterize a scene dates back to the early 2000s [34, 35]. This technique models a 3D scene with a collection of 3D Gaussian reconstruction kernels parameterized by 59 parameters representing location, scale, orientation, color, and opacity of the kernel. Initially, each Gaussian is derived from a point in a Structure from Motion (SfM) reconstruction. These parameters are subsequently refined through a dual process involving the minimization of image rendering loss and adaptive kernel density adjustment. The efficiency of 3D-GS is enhanced by a GPU-optimized, tile-based rasterization method, enabling real-time rendering of complex 3D scenes. Employing 3D Gaussians as reconstruction kernels simplifies the volumetric rendering process and facilitates the integration of a low-pass filter within the kernel. This

(a) Five Gaussians fitting a square

(b) Three HGs fitting a square

(c) Gaussian and Half-Gaussian in spatial domain

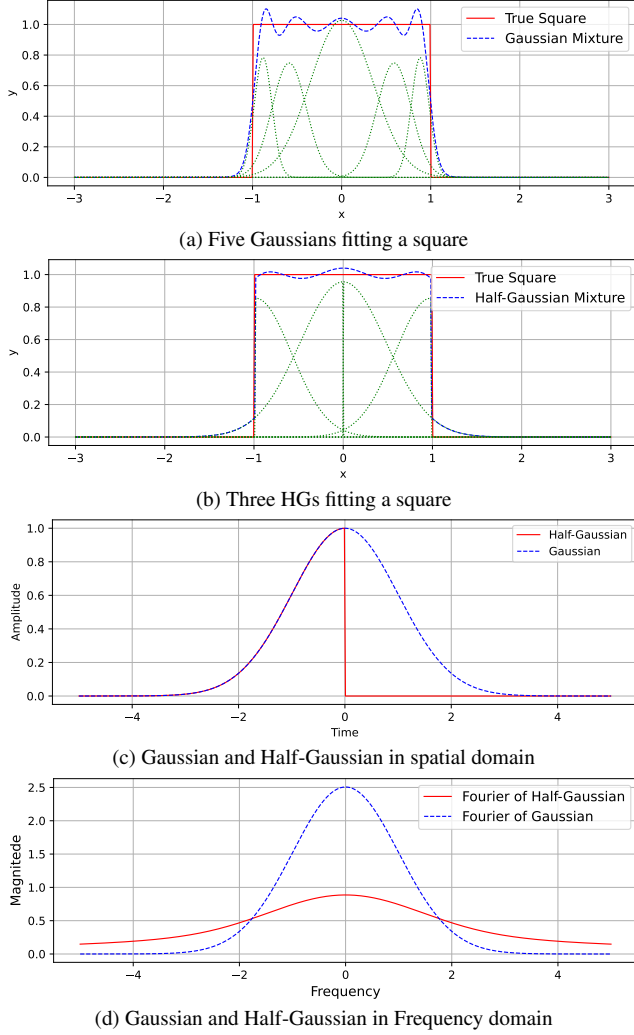(d) Gaussian and Half-Gaussian in Frequency domain

Figure 2. **Comparison of Half-Gaussian and Gaussian Kernels fitting a square function and their Fourier Transforms.** (a): fitting a square function with 5 Gaussian kernels, and (b): fitting a square with 4 Half-Gaussian kernels. When approximating sharp edges, the Half-Gaussian kernels achieve a lower error loss (1.85) compared to Gaussian kernels (2.97). Figures (c) and (d) illustrate the Gaussian and Half-Gaussian kernels in both the spatial and frequency domains, where the Half-Gaussian demonstrates a higher bandwidth than the Gaussian kernel, indicating its superior ability to capture high-frequency components.

addition effectively mitigates aliasing during the resampling of scene data to screen space [33, 35].

While 3D-GS benefits from employing 3D Gaussians, using a Gaussian kernel can be inefficient and lead to inaccuracies when modeling discontinuous functions, which are common in 3D scenes at object boundaries and texture-rich areas. As Fig. 2(a) shows, using Gaussian kernels leads to a Gibbs phenomenon with high peaks overshooting and undershooting the function value and non-vertical transitions
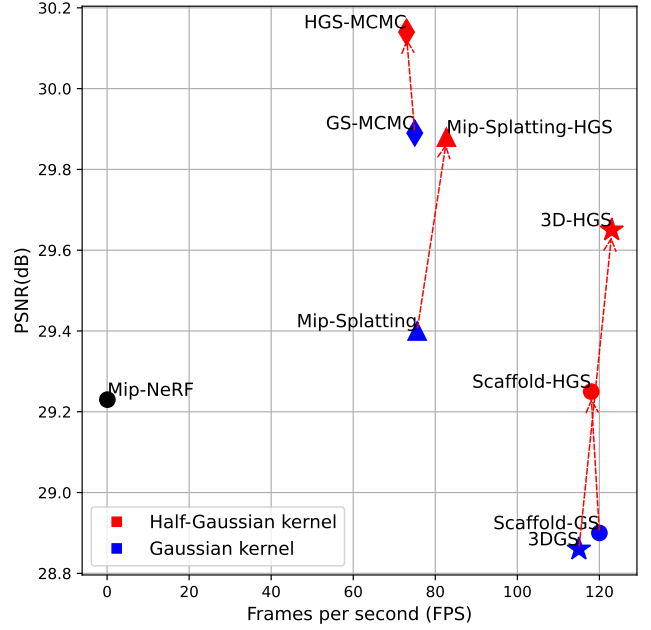

Figure 3. Performance (PSNR↑) versus rendering speed for several state-of-the-art methods [1, 9, 10, 15, 29] with Gaussian kernels and the proposed half-Gaussian kernels on the Mip-NeRF360 dataset [1]. In all cases, using half-Gaussian kernels resulted in significant PSNR improvements, with similar or better rendering speed than the corresponding 3D Gaussian-based method.

(i.e. blurry edges). To address this problem, we propose the use of 3D-Half-Gaussian Splatting (3D-HGS), which employs a 3D-Half Gaussian as a novel reconstruction kernel. As seen in Fig. 2(b), using this kernel significantly reduces the Gibbs oscillatory effect and better fits the vertical discontinuities. This can be explained by the fact that Half-Gaussians have a higher content at higher frequencies than full Gaussians do (Fig. 2 (c) and (d)).

A pair of 3D half-Gaussians can be easily represented by adding the parameters of a vector normal to a plane splitting a 3D Gaussian through its center, and allowing each half to have different opacity values (Fig. 4(b)). By introducing this plane, the kernel can efficiently capture high-frequency information at discontinuities, while preserving the essential characteristics of the original 3D Gaussian kernel (Fig. 2). This preservation is facilitated by the symmetric pairing of 3D Half-Gaussians, which also allows to seamlessly represent full 3D Gaussians (see for example the center Gaussian in Fig. 2(b)).

Thus, the proposed 3D-Half-Gaussian kernel preserves the key parameters in the original 3D-GS and provides the capability to be easily applied to existing 3D Gaussian kernel-based methods as a plug-and-play kernel. Our experiments (see Fig. 3 and section 4.2) show that using the proposed 3D Half-Gaussian as the reconstruction ker-

nel achieves state-of-the-art (SOTA) rendering quality performance on MipNeRF-360, Tanks&Temples, and Deep Blending datasets, with similar or better rendering speed than the corresponding 3D Gaussian-based methods.

Our main contributions are: (1) We introduce a novel plug-and-play reconstruction kernel, the 3D Half-Gaussian, designed to enhance the performance of 3D Gaussian Splatting (3D-GS). (2) Our proposed kernel achieves state-of-the-art novel view synthesis performance across multiple datasets without compromising rendering frame rate. (3) We demonstrate the versatility and effectiveness of our kernel by applying it to other state-of-the-art methods, showcasing its broad applicability. Our code will be available on our GitHub project page.

2. Related Work

3D reconstruction and Novel View Synthesis (NVS) have long been key goals in computer vision. NeRFs [17] have significantly advanced NVS, enabling highly realistic image synthesis from new viewpoints. More recently, 3D-GS has set new state-of-the-art benchmarks in this field. This section reviews both historical and recent developments in 3D reconstruction and NVS, with a detailed analysis of 3D-GS, its methods, achievements, and impact on the field.

2.1. Novel View Synthesis

Before the advent of NeRFs, Multi-View Stereo (MVS) [20] and Structure from Motion (SfM) were commonly utilized for reconstructing 3D scenes from multiple viewpoint images. MVS relies on feature extraction from diverse images to correlate viewpoints and produce a final reconstruction, typically represented as a colored mesh grid or point cloud. However, due to its reliance solely on image features, achieving high-quality reconstructions can be challenging. SfM [19], employs multi-view images to generate a point cloud representing the 3D scene. In contrast to MVS, SfM excels in accurately estimating camera poses for different images, while MVS provides more precise 3D estimations of the scene. Notably, SfM's simultaneous estimation of camera positions and point clouds makes it a preferred preprocessing step in recent advanced NVS methods.

NeRFs [17], stand as a significant milestone in NVS, demonstrating remarkable achievements in tasks such as image and view synthesis alongside scene reconstruction. Unlike previous methods, a Nerf represents the 3D scene using a radiance field, treating the entire scene as a continuous function parameterized by position.

3D-GS [9], is a recently introduced NVS technique. This method boasts a remarkable reduction in both training and rendering times, surpassing the Nerf methods. Diverging from its predecessors, 3D-GS does not rely on training neural networks or any type of network architecture. Instead, it initiates from a point cloud. Unlike treating each point

discretely, 3D-GS conceptualizes them as 3D Gaussian entities, each possessing a unique size and orientation, and spherical harmonics to depict their color. Subsequently, during the splatting stage, these 3D Gaussians are projected onto a 2D plane, with their appearances accumulated to generate renderings for a given viewing angle.

2.2. Splatting methods

The original concept of splatting was introduced by Westover [25, 26], and improved by Zwicker et. al [33–35]. Recently, the 3D-GS technique has achieved great success in photo-realistic neural rendering [9, 27].

Although 3D-GS has attained state-of-the-art performance in terms of rendering speed and quality, opportunities for further enhancements remain. Various studies [2, 3] have introduced modifications to the original framework. For instance, Mip-splatting [29] limits the frequency of the 3D representation to no more than half the maximum sampling frequency, as determined by the training images. Analytic-Splatting [13] employs a logistic function to approximate the Gaussian cumulative distribution function, thus refining each pixel's intensity response to minimize aliasing. Similarly, SA-GS [21] adapts during testing a 2D low-pass filter based on rendering resolution and camera distance. Scaffold-GS [15] employs voxel grids for initializing 3D scenes and utilizes Multi-Layer Perceptrons (MLPs) to constrain and learn voxel features. Following a similar paradigm, SAGS [23] incorporates a graph neural network to capture structural relationships between individual Gaussians, preserving geometric consistency across neighboring regions during rendering. FreGS [30] introduces frequency-domain regularization to the rendered 2D images to enhance the recovery of high-frequency details. 2D-GS [8] aligns 3D scenes with 2D Gaussian kernels to improve surface normal representations. MCMC-GS [10] applied Stochastic Gradient Langevin Dynamics (SGLD) to iteratively refine Gaussian positions. Lastly, GES [6] employs generalized exponential kernels to reduce the memory required to store 3D information.

The two papers most closely related to ours are 2D-GS [8] and GES [6], as they also focus on switching the reconstruction kernels. However, GES did not change the rasterizer but learned a scaling factor for each of the points to approximate different kernels. This method faces difficulties in complex 3D scenes. On the other hand, instead of doing a volumetric rendering as in the original 3D-GS, 2D-GS tries to do a surface rendering. Our method still learns a volumetric rendering of scenes to retain the rendering performance while enabling accurate modeling of discontinuous functions.
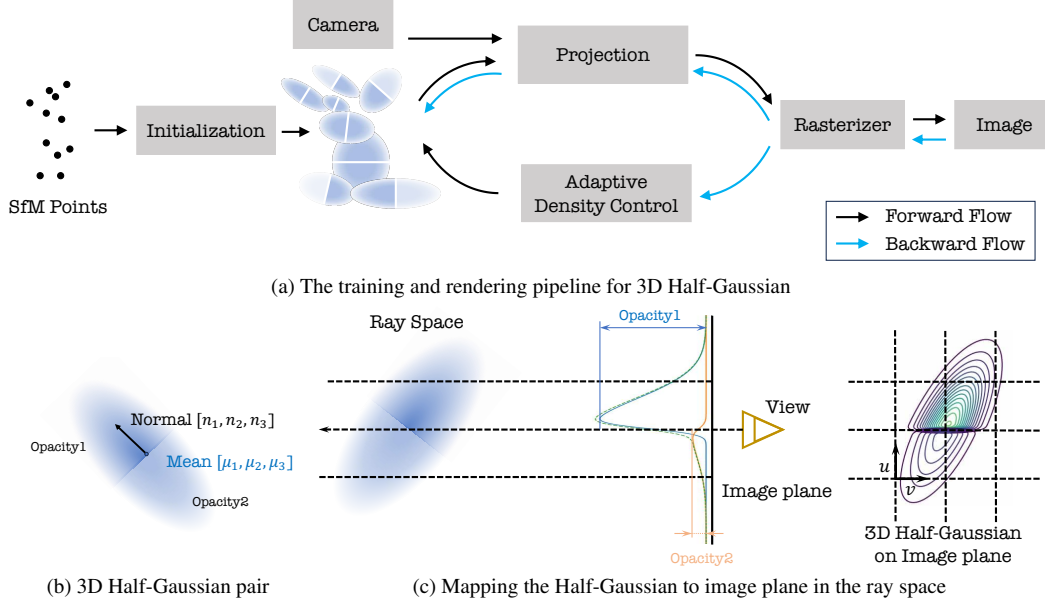
(a) The training and rendering pipeline for 3D Half-Gaussian

(b) 3D Half-Gaussian pair   (c) Mapping the Half-Gaussian to image plane in the ray space

Figure 4. Illustration of the 3D-HG kernel, and the mapping of a pair of 3D Half-Gaussians to a 2D image.

3. Method

3D-GS [9] models 3D scenes using 3D Gaussian kernels [34, 35]. However, these kernels frequently encounter difficulties in accurately modeling 2D/3D discontinuous functions, which are prevalent in 3D scenes at edges, corners of objects, and texture-rich areas. The inefficiencies of the 3D Gaussian kernel in representing shape and color discontinuities can compromise the model's effectiveness. To overcome these limitations, we propose the use of a Half-Gaussian kernel for novel view synthesis as illustrated in Fig. 4. Section 3.1 presents an overview of the foundational concepts relevant to 3D Gaussian Splatting. In Section 3.2, we introduce the 3D half-Gaussian kernel, followed by an in-depth description of the 3D-HGS rasterizer in Section 3.3. Finally, Section 3.4 provides a detailed account of the splatting process.

3.1. Preliminaries

3D-GS [9] represents the 3D scenes with parameterized antisotropic 3D Gaussian kernels [34, 35]. It starts from an initial set of 3D Gaussians located at a set of sparse points obtained using a Structure-from-Motion (SfM) step. Then, 3D Gaussians are mapped to 2D images through a GPU-specified tile-based rasterization. The parameters of these Gaussians are optimized, pruned, and added based on a loss function on the rendered images and ground-truth images.

Each of the 3D Gaussians is parameterized by their position (mean) μ, covariance-related scaling matrix, opacity α, and Spherical harmonics coefficients for color c. A 3D elliptical Gaussian $G_{\mathbf{\Sigma}}(\mathbf{x})$ centered at a point μ with covari-

ance matrix $\mathbf{\Sigma}$ is given by:

$$G_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = \frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \quad (1)$$

where $\mathbf{x}$ and μ are the column vectors $[x, y, z]^T$ and $[\mu_x, \mu_y, \mu_z]^T$, respectively, and $\mathbf{\Sigma}$ is a positive definite 3×3 matrix. The covariance under the world coordinate system $\mathbf{\Sigma}^w$ is further parameterized by a scaling matrix S and a rotation matrix R to maintain its positive definite property:

$$\mathbf{\Sigma^w} = \boldsymbol{RSS^T R^T} \quad (2)$$

Given a viewing transformation W and the Jacobian of the affine approximation of the projective transformation J, the covariance matrix $\mathbf{\Sigma}$ in the camera coordinate system is given by:

$$\mathbf{\Sigma} = JW\mathbf{\Sigma}^w W^T J^T \quad (3)$$

As in the rendering function we will learn the opacity parameter α, we can merge the constant term $\frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}|^{\frac{1}{2}}}$ into α. Integrating a normalized 3D Gaussian along one coordinate axis results in a normalized 2D Gaussian. Thus, 3D Gaussians $G(\mathbf{x})$ can be efficiently transformed to 2D Gaussians $\hat{G}(\hat{\mathbf{x}})$ on the image plane using a ray coordinate system representation [33]:

$$\int_{\mathbb{R}} G_{\mathbf{\Sigma}}(\mathbf{x} - \mu)dz = \hat{G}_{\hat{\mathbf{\Sigma}}}(\hat{\mathbf{x}} - \hat{\mu}) \quad (4)$$

where $\hat{x} = [x, y]^T$, $\hat{\mu} = [\mu_x, \mu_y]^T$, and the covariance $\hat{\mathbf{\Sigma}}$ can be easily obtained by taking the top-left 2×2 sub-matrix of the transformed $\mathbf{\Sigma}$:

$$\boldsymbol{\Sigma} = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \Leftrightarrow \begin{pmatrix} a & b \\ b & d \end{pmatrix} = \hat{\boldsymbol{\Sigma}} \qquad (5)$$

Finally, the pixel values on the 2D image are obtained by α-blending:

$$C = \sum_{i \in \mathcal{N}} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \ \ \sigma_i = \alpha_i \hat{G}(\hat{\boldsymbol{x}} - \hat{\boldsymbol{\mu}}) \qquad (6)$$

where $\hat{\mathbf{x}}$ is the queried pixel position and $\mathcal{N}$ is the set of sorted 2D Gaussians associated with $\hat{\mathbf{x}}$.

3.2. 3D Half-Gaussian Kernel

In this section, we provide a detailed description of the proposed kernel. We begin by giving the definition of the half Gaussian kernel and how to parameterize it.

We propose to use 3D Half-Gauusians kernels, where each half can have different opacity values α_1 and α_2. The half Gaussians are obtained by splitting a 3D Gaussian with a plane through its center. Note that this representation includes as a special case the 3D-GS representation in the case when $\alpha_1 = \alpha_2$. However, incorporating a planar surface into the kernel allows to represent sharp changes, such as edges and textures, as well as planar surfaces, more accurately. As seen below, the increase in the number of learned parameters is very modest: the 3D normal of the splitting plane and an additional opacity term. Furthermore, it should be noted that the plane normal can be stored using the normal field, which is available but not used, in the original 3DGS kernel implementation. Thus, 3DHGS effectively increases the memory requirements only by an extra opacity coefficient, while it increases computational cost by a multiplication by a scaling factor.

Formally, a 3D Half-Gaussian kernel is described by:

$$HG_{\boldsymbol{\Sigma}}(\mathbf{x}-\mu) = \begin{cases} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} & \mathbf{n}^T(\mathbf{x}-\mu) \geq 0 \\ 0 & \mathbf{n}^T(\mathbf{x}-\mu) < 0 \end{cases} \qquad (7)$$

where $\mathbf{n} = [n_1, n_2, n_3]^T$ is the normal of the splitting plane through the full Gaussian center, represented as a column vector, while the complementary half Gaussian is obtained by simply changing the sign of the normal. The integration of a 3D Half Gaussian along the z axis yields a similar result as Eq. 4, except for a scaling factor related to the normal of the splitting plane:

$$\int_{\mathbf{n}^T(\mathbf{x}-\mu)\geq 0} HG_{\boldsymbol{\Sigma}}(\mathbf{x}-\mu)dz = \frac{1}{2}I(x,y)\hat{G}_{\hat{\boldsymbol{\Sigma}}}(\hat{\mathbf{x}}-\hat{\mu}) \quad (8)$$

where

$$I(x,y) = \text{erfc}\left(-\frac{(n_1 x + n_2 y)/n_3 + \mu_{z|xy}}{\sqrt{2}\sigma_{z|xy}} \right) \qquad (9)$$
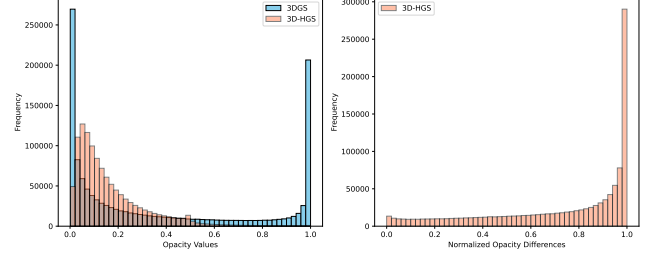

Figure 5. (Left) Histograms of the 3DGS opacity values and of the 3D-HGS mean opacity values of corresponding halves, trained on the Bonsai scene. The 3D-HGS mean opacity values cluster in a lower range than those in the 3DGS, implying that treating both halves identically while rendering increases the number of kernels involved in each tile, slowing down the overall process. (Right) Histogram of the differences between opacity values within a half Gaussian, normalized by the maximum opacity value within each kernel. Over 75% of the 3D-HGS kernels have normalized opacity differences over 0.5, highlighting that each half Gaussian often represents a distinct effective area in the rendering space.

and $\mu_{z|xy}$, and $\sigma_{z|xy}$ refer to the mean and standard deviation for the distribution of z, conditioned on given x, y, i.e. $P(z|x,y) \sim \mathcal{N}(\mu_{z|xy}, \sigma_{z|xy})$, respectively.

Eq. 8 and Eq. 9 give a closed-form solution to calculate the integral of a 3D Half-Gaussian. The right-hand side of Eq. 8 consists of two factors: $I(x,y)$, which is a scaling factor related to the learned normal of the splitting plane, while $\hat{x} - \hat{\mu}$ remains the same as in the original 3D Gaussian in Eq. 4. For a detailed derivation please refer to the supplementary material.

3.3. 3D Half-Gaussian Rasterization

For rasterization, we follow a similar process as in 3D-GS, adapted to the proposed 3D Half-Gaussian reconstruction kernel. For enhanced parameter efficiency, we jointly represent pairs of Half-Gaussians, since the parameters for mean, rotation, scaling, and color are shared between the two halves, while we learn the orientation of the splitting plane and two opacity parameters, one for each side of the splitting plane. Thus, the volumetric alpha blending for each pixel on the image can be expressed as :

$$C = \sum_{i \in \mathcal{N}} c_i H\hat{G}_i(\hat{\mathbf{x}} - \hat{\mu}) \prod_{j=1}^{i-1} \left(1 - H\hat{G}_j(\hat{\mathbf{x}} - \hat{\mu}) \right) \quad (10)$$

where $\mathcal{N}$ is the sorted 3D Half-Gaussian set for the pixel and $H\hat{G}(x)$ is the integration of both halves of a 3D Half-Gaussian pair:

$$H\hat{G}(\hat{\mathbf{x}} - \hat{\mu}) = \frac{1}{2}\left\{ 2\alpha_2 + (\alpha_1 - \alpha_2)I(x,y) \right\} \hat{G}_{\hat{\boldsymbol{\Sigma}}}(\hat{\mathbf{x}} - \hat{\mu}) \quad (11)$$
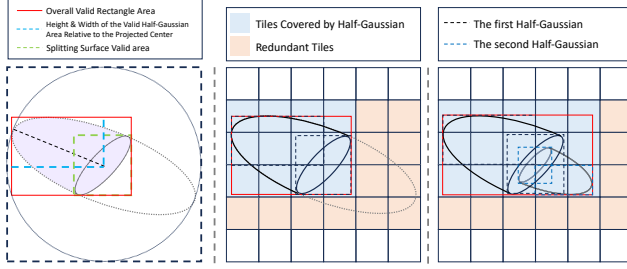
Figure 6. Left: Method for calculating the bounding rectangle of a half-Gaussian. Middle: Visualization when one half of the Gaussian is transparent. Right: Visualization when each side of the Gaussian has distinct opacity levels.

3.4. 3D Half-Gaussian Splatting

The 3D half-Gaussian kernel has a splitting plane along with distinct opacity values for each half (Fig. 4b). Unlike conventional splatting methods – which compute the projection shape of a Gaussian as a whole – the half-Gaussian kernel requires specialized handling.

Naively applying standard splatting uniformly across both halves is inefficient since, in general, a large number of kernels have one of their halves fully transparent. This is supported by Fig. 5, where the left plot shows that the average opacity values per half-Gaussian kernel cluster in a lower range than when using the full Gaussian kernel and the right plot shows that over 75% of the kernels have normalized opacity differences greater than 0.5. Therefore, treating both halves identically while rendering would increase the number of kernels involved in each tile, slowing down the overall process.

Thus, we propose an efficient 3D half-Gaussian splatting technique that independently calculates the valid region for each half as illustrated in Fig. 6. Initially, we project the ellipsoidal splitting surface onto the 2D image plane, defining the inner rectangle (relative to the projected center) of the projected half-Gaussian (green dashed rectangle in Fig. 6). To determine the outer bound, inspired by [4], we compute the height and width of the tangent edges relative to the ellipse's center (blue dashed lines in Fig. 6), thereby establishing the limits of the valid region. This region is encapsulated by the minimal bounding rectangle that fully encloses the projected splitting ellipse along with its tangent edges (red rectangle in Fig. 6).

In practice, if one side of the half-Gaussian is entirely transparent, only the opaque half is splatted, as depicted in Fig. 6, center. When both halves contribute, the valid region is determined by the outer bounding tangent edges and the projected splitting ellipse Fig. 6, right.

4. Experiments

4.1. Experimental Setup

Datasets and Metrics. Following the published literature, we tested our design on 11 (indoor and outdoor) scenes from various datasets: 7 scenes from Mip-nerf360 dataset[1], 2 scenes from Tanks &Temples [11], and 2 scenes from DeepBlending [7].

Consistent with prior studies, we use PSNR, SSIM [24] and LPIPS [31] to measure the performance on each dataset. We provide the averaged metrics over all scenes for each dataset in the main paper and give the full quantitative results for each scene in the Appendix. We also report rendering times and model size.

Baselines. To evaluate the general improvement brought by using a Half-Gaussian kernel in neural rendering, we ran experiments based on four baseline methods: vanilla-3D-GS [9] and three of its extensions Scaffold-GS [15], Mip-Splatting [29], and GS-MCMC [10], where we replaced the reconstruction kernel with the Half-Gaussian kernel. We denote our models as 3D-HGS, Scaffold-HGS, Mip-Splatting-HGS, and HGS-MCMC, respectively. We compared performance against state-of-the-art 3D reconstruction methods: 3D-GS [9], Mip-NeRF [1], 2D-GS [8], Fre-GS [30], Scaffold-GS [15], GES [6], Mip-Splatting [29], and GS-MCMC [10].

Reconstruction Kernels. We compared the rendering performance of the half-Gaussian kernel against three other reconstruction kernels: the original 3D-GS kernel, the 2D Gaussian kernel (2D-GS) and the generalized exponential kernels (GES), which were proposed in [8] and [6], respectively. For a fair comparison, we used the same loss function and training iterations as the original 3D-GS. For details on the implementation of different kernels, please refer to the supplementary material.

Implementation. For our implementation, we utilize the three (unused) normal parameters in 3D-GS to represent the normal vector of the splitting plane. Additionally, we learn one opacity for each half of the Gaussian. This results in memory increasing by only one additional parameter for each reconstruction kernel compared to the 3D-GS. The forward and backward passes of the rasterizer are modified based on the vanilla 3D-GS and Eq. 11. For 3D-HGS and Mip-Splatting-HGS, we adhered to the training settings and hyperparameters used in [9], and [29], setting the learning rate for the normal vector at 0.3 for the Deep Blending dataset and 0.003 for the other datasets. For the implementation of Scaffold-HGS, we did not increase the number of parameters in each Gaussian. Following the referenced paper, we employed an MLP to learn the normal vector based on the feature vector of each voxel and doubled the width of the output layer of the opacity MLP to accommodate two opacity values. The rasterizer was also modified based on

Table 1. **Quantitative comparison to the SOTA methods on real-world datasets.**

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Method	PSNR ↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [1]	29.23	0.844	0.207	22.22	0.759	0.257	29.40	0.901	0.245
2D-GS[8]	28.98	0.867	0.185	23.43	0.845	0.181	29.70	0.902	0.250
Fre-GS [30]	27.85	0.826	0.209	23.96	0.841	0.183	29.93	0.904	0.240
GES [6]	28.69	0.857	0.206	23.35	0.836	0.198	29.68	0.901	0.252
3D-GS [9]	28.88	0.870	0.182	23.60	0.847	0.181	29.41	0.903	0.243
3D-**HGS** (**Ours**)	29.66 +0.78	0.873	0.178	24.45 +0.85	0.857	0.169	29.76 +0.35	0.905	0.242
Scaffold-GS [15]	28.95	0.848	0.220	23.96	0.853	0.177	30.21	0.906	0.254
Scaffold-**HGS**(**Ours**)	29.25 +0.30	0.867	0.186	24.42 +0.46	0.859	0.162	30.36 +0.15	0.910	0.240
Mip-Splatting [29]	29.39	0.880	0.162	23.75	0.857	0.157	29.46	0.903	0.243
Mip-Splatting-**HGS**(**Ours**)	29.88 +0.49	0.881	0.160	24.53 +0.78	0.865	0.145	29.61 +0.15	0.901	0.241
GS-MCMC [10]	29.89	0.900	0.190	24.29	0.860	0.190	29.67	0.890	0.320
HGS-MCMC(**Ours**)	30.13 +0.24	0.886	0.158	25.08 +0.77	0.841	0.144	29.80 +0.13	0.898	0.245

Eq. 11. For HGS-MCMC, we adopt the hyperparameters and training settings from [10], with the exception that we set the opacity threshold for Gaussian relocation to 0.015 for the Playroom scene. Additional training details are provided in the supplementary material. All experiments were conducted using an NVIDIA RTX 3090 GPU.
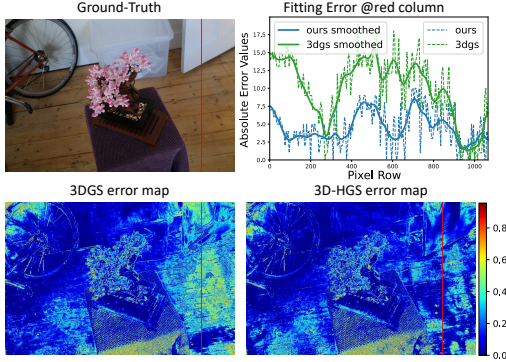

Figure 7. **Fitting Error.** Top-left: Frame from Bonsai scene. Top-right: Fitting error using 3DGS and 3DHGS, along the red column. Bottom: Fitting error on the entire image, shown as a normalized heat map in the range [0,1] for 3DGS and 3DHGS.

4.2. Results Analysis

Quantitative results are presented in Figs. 3 and 7, and in Tabs. 1 and 2. Further details are provided in the supplementary material. Across all datasets and methods, the integration of the 3D Half-Gaussian kernel yields substantial gains, establishing it as a superior choice for enhancing 3D novel view synthesis accuracy.

Fig. 7 shows plots of the grayscale fitting for the Bonsai scene along an image column containing both smooth surface regions and sharp transitions, along with the corresponding fitting error map for the entire image. The plots compare the proposed half-Gaussian kernel and standard 3D Gaussian Splatting (3DGS) against the ground truth. Consistent with the results in Fig. 2, they illustrate that

the half-Gaussian kernel achieves a more accurate fit than 3DGS, especially in areas with sharp transitions.

Table 1 shows that incorporating the proposed Half Gaussian kernel, 3D-HGS, Scaffold-HGS, Mip-Splatting-HGS, and MCMC-HGS achieved SOTA performance across multiple datasets, consistently outperforming their respective baselines. This improvement underscores the effectiveness of the 3D Half-Gaussian kernel as a robust choice for the novel view synthesis. On the MipNeRF360 dataset, our proposed kernel enables 3D-HGS, Scaffold-HGS, Mip-Splatting-HGS, and MCMC-HGS to surpass baseline PSNRs by 0.78, 0.30, 0.49, and 0.24, respectively, with MCMC-HGS achieving a new state of the art. For the Tanks and Temples dataset, these methods see PSNR improvements of 0.85, 0.46, 0.78, and 0.77, again with MCMC-HGS setting a new benchmark. Lastly, on the Deep Blending dataset, our methods outperform baselines by 0.35, 0.15, 0.15, and 0.13 PSNR, with Scaffold-HGS achieving state-of-the-art performance.

Table 2. **Comparison of rendering speed and storage memory** between the proposed Half-Gaussian kernel-based methods and traditional Gaussian kernel-based methods.

Dataset	Mip-NeRF360		Tanks&Temples		Deep Blending	
Method	FPS ↑	Mem ↓	FPS ↑	Mem ↓	FPS ↑	Mem ↓
3D-GS [9]	115	762	149	429	104	668
3D-**HGS** (**Ours**)	125	694	160	437	126	641
Scaffold-GS [15]	120	173	120	77	129	55
Scaffold-**HGS**(**Ours**)	118	180	115	84	136	53
GS-MCMC [10]	75	732	133	438	90	969
HGS-MCMC(**Ours**)	72	743	139	445	92	980
Mip-Splatting [10]	76	970	117	569	91	843
Mip-Splatting-**HGS**(**Ours**)	83	883	121	566	102	808

Finally, using half-Gaussian kernels does not significantly impact the rendering speed and memory requirements, as shown in Fig. 3, Table 2, and in the ablation study in the supplemental material.

Qualitative results are shown in Figs. 8, 9, and in the

Figure 8. **Qualitative comparison between 3D-GS and 3D-HGS.**

Table 3. **PSNR scores ↑ of the rendering performance with different reconstruction kernels.** Red indicates performance improvement, while blue denotes performance decline.

| Kernels | Tanks&Temples | | | Deep Blending | | | Mip-NeRF360 | | | | | | | | AVG |
	Truck	Train	Avg	Playroom	Drjohnson	Avg	Bicycle	Garden	Kitchen	Stump	Room	Counter	Bonsai	Avg	
3D-GS	25.30	21.91	23.60	29.98	29.09	29.53	25.21	27.30	30.80	26.56	31.40	28.70	32.20	28.88	28.04
2D-GS	25.14	21.70	23.42	30.18	29.12	29.65	25.02	27.14	31.33	26.57	31.37	28.97	32.33	28.94	28.06
GES	24.94	21.73	23.34	30.29	29.35	29.82	24.87	27.07	31.07	26.17	31.17	28.75	31.97	28.72	27.94
3D-**HGS**	26.25	22.65	24.45	30.20	29.30	29.76	25.25	27.50	32.22	26.64	32.52	29.84	33.52	29.66	28.70
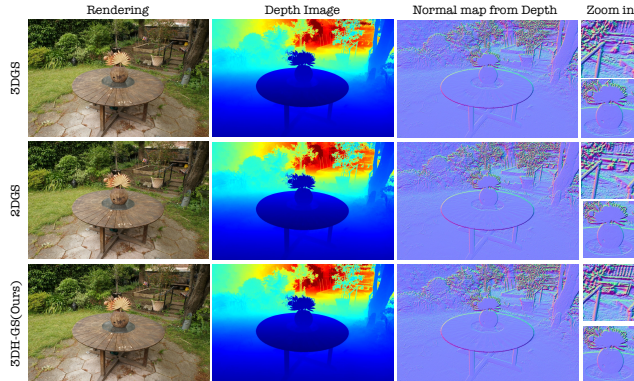
Figure 9. **Comparison of Depth Images and Normal Maps.** We visualize the depth maps alongside normals estimated from these maps. Our method demonstrates superior performance, capturing finer details in the bench structure and the surface textures of pots.

supplemental material. In Fig. 8 it can be observed that our method delivers better performance on fine-scale details (e.g., T&T-Truck, Mip360-Garden, Mip360-Bonsai, Mip360-Room), high-frequency textures (e.g., T&T-Train, Mip360-Counter), light rendering (e.g., Mip360-Garden, DB-DrJohnson), and shadow areas (e.g., T&T-Train, Mip360-Bonsai, DB-DrJohnson). Fig. 9 provides an example of a generated depth image and the corresponding estimated surface normals, which were produced using the method described in [8]. The proposed method captures

better finer details in the bench structure and the surface texture of the pots.

Different Reconstruction Kernels Table 3 provides the PSNR score for each 3D scene, the average score for each dataset, and the total average score when using the different kernels. In this experiment, we use 3D-GS as the baseline, with improved and degraded results highlighted in red and blue, respectively. Compared to the original 3D Gaussian kernel, the 3D Half-Gaussian kernel shows consistent improvement across all 3D scenes. While other kernels demonstrate superiority in some 3D scenes, they exhibit drawbacks in others. Overall, we achieved the best average performance across all 3D scenes.

5. Conclusion

We introduced Half-Gaussian Splatting, a plug-and-play method for accurate 3D view synthesis. Our approach uses two distinct opacities in each Gaussian, allowing precise rendering control without increasing inference time. This design achieves SOTA performance across multiple datasets and integrates seamlessly into most Gaussian-splatting architectures without structural changes. We validated the effectiveness of our approach across four baseline methods, consistently improving accuracy. To rigorously assess its advantages, we compared it against other kernel modification techniques, confirming Half-Gaussian Splatting as a highly effective choice for 3D splatting-based methods.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2, 6, 7

[2] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024. 3

[3] Anurag Dalal, Daniel Hagen, Kjell G Robbersmyr, and Kristian Muri Knausgård. Gaussian splatting: 3d reconstruction and novel view synthesis, a review. *arXiv preprint arXiv:2405.03417*, 2024. 3

[4] Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Zhilin Pei, Hengjie Li, Xingcheng Zhang, and Bo Dai. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. *arXiv preprint arXiv:2408.07967*, 2024. 6

[5] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 1

[6] Abdullah Hamdi, Luke Melas-Kyriazi, Guocheng Qian, Jinjie Mai, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. *arXiv preprint arXiv:2402.10128*, 2024. 3, 6, 7

[7] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 6

[8] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024. 3, 6, 7, 8

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 1, 2, 3, 4, 6, 7

[10] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3, 6, 7

[11] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 6

[12] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. *arXiv preprint arXiv:2311.16096*, 2023. 1

[13] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. *arXiv preprint arXiv:2403.11056*, 2024. 3

[14] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. *arXiv preprint arXiv:2402.17427*, 2024. 1

[15] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2, 3, 6, 7

[16] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 1

[17] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 3

[18] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. 1

[19] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3

[20] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, pages 519–528. IEEE, 2006. 3

[21] Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. Sags: Scale-adaptive gaussian splatting for training-free anti-aliasing. *arXiv preprint arXiv:2403.19615*, 2024. 3

[22] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022. 1

[23] Evangelos Ververas, Rolandos Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. Sags: Structure-aware 3d gaussian splatting. *arXiv preprint arXiv:2404.19149*, 2024. 3

[24] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[25] Lee Westover. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, pages 9–16, 1989. 3

[26] Lee Westover. Footprint evaluation for volume rendering. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, 1990. 3

[27] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *arXiv preprint arXiv:2403.11134*, 2024. 3

[28] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024. 1

[29] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2, 3, 6, 7

[30] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024. 3, 6, 7

[31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[32] Cheng Zhao, Su Sun, Ruoyu Wang, Yuliang Guo, Jun-Jun Wan, Zhou Huang, Xinyu Huang, Yingjie Victor Chen, and Liu Ren. Tclc-gs: Tightly coupled lidar-camera gaussian splatting for surrounding autonomous driving scenes. *arXiv preprint arXiv:2404.02410*, 2024. 1

[33] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 2, 3, 4

[34] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 1, 4

[35] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 1, 2, 3, 4