Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration

Zhihao Liang[1,*], Qi Zhang[2,†], Wenbo Hu[2],
Lei Zhu[3], Ying Feng[2], and Kui Jia[4,†]

[1]South China University of Technology
[2]Tencent AI Lab [3]City University of Hong Kong
[4]School of Data Science, The Chinese University of Hong Kong, Shenzhen
https://lzhnb.github.io/project-pages/analytic-splatting/

Abstract. 3D Gaussian Splatting (3DGS) recently gained popularity by combining the advantages of both primitive-based and volumetric 3D representations, resulting in improved quality and efficiency for 3D scene rendering. However, 3DGS is not alias-free and still produces severe blurring or jaggies when rendered at varying resolutions because the discrete sampling scheme used treats each pixel as an isolated single point, which is insensitive to changes in the footprints of pixels and is restricted in sampling bandwidth. In this paper, we use a conditioned logistic function as the analytic approximation of the cumulative distribution function (CDF) of the Gaussian signal and calculate the integral by subtracting the CDFs. We introduce this approximation to two-dimensional pixel shading and present *Analytic-Splatting*, which analytically approximates the Gaussian integral within the 2D-pixel window area to better capture the intensity response of each pixel. Then, we use the approximated response of the pixel window integral area to participate in the transmittance calculation of volume rendering, making Analytic-Splatting sensitive to the changes in pixel footprint at different resolutions. Extensive experiments on various datasets validate that our approach has better anti-aliasing capability that gives more details and better fidelity.

Keywords: 3D Gaussian Splatting · Anti-Aliasing · View Synthesis · Cumulative Distribution Function (CDF) · Analytic Approximation

1 Introduction

Novel view synthesis of a scene captured from multiple images has achieved great progress due to the rapid advancements of neural rendering. As a prominent representative, Neural Radiance Field (NeRF) [25] models the scene using a neural volumetric representation, enabling photorealistic rendering of novel

* Work was done during an internship at Tencent AI Lab.
† Corresponding authors.

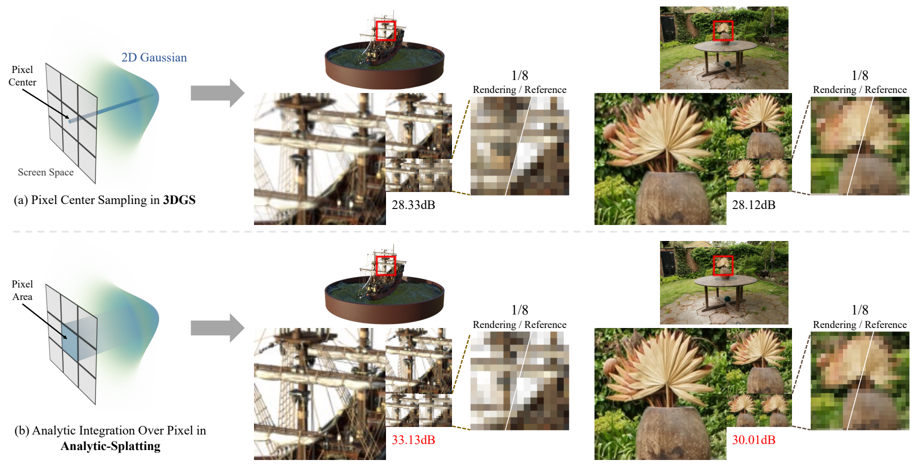


Fig. 1: For shading a pixel by a Gaussian signal, 3DGS (a) only treats the Gaussian signal value corresponding to the pixel center as the intensity response. Analytic-Splatting (b) instead considers an analytic approximation of the integral over the pixel window area as the intensity response. Compared to 3DGS, Analytic-Splatting has anti-aliasing capability and better detail fidelity.

views via ray casting. Ray casting trades off efficiency with quality, and subsequent works [9, 26, 34] are proposed to have a better quality-efficiency balance. Recently, 3D Gaussian Splatting (3DGS) [18] proposes a GPU-friendly differentiable rasterization that incorporates explicit primitives, achieving high-quality and real-time renderings for novel view synthesis. In contrast to the ray casting in NeRF, which renders a pixel by accumulating the radiance of samples along the ray that intersects the image plane at the pixel, 3DGS employs a rasterization-based technique that can be rasterized very efficiently. Specifically, 3DGS represents the scene as a set of anisotropic 3D Gaussians with scene properties; when rendering a pixel, 3DGS orders and projects these 3D Gaussians onto the image plane as 2D Gaussians, and then queries values and scene properties associated with the Gaussians that have overlaps with the pixel, and finally shades the pixel by cumulatively compositing these queried values and properties.

3DGS works for scene representation learning and novel view synthesis at constant resolutions; however, its performance degrades greatly either when the multi-view images are captured at varying distances, or when the novel view to be rendered has a resolution different from those of the captured images. The main reason is that the *footprint* of the pixel changes at different resolutions. Specifically, footprint denotes the ratio between the pixel window area in screen space and its covered Gaussian signals region in the world space. Besides, 3DGS is insensitive to such changes since it treats each pixel as an isolated point (*i.e.* merely pixel center) when retrieving the corresponding Gaussian values; Fig. 1a gives an illustration. As a result, 3DGS could produce significant artifacts (*e.g.* blurry or jaggies) especially when pixel footprints change drastically (*e.g.* synthesizing novel views with zooming-in and zooming-out effects).

In 3DGS, a continuous signal in the image space is represented as a set of α-blended 2D Gaussians, and the pixel shading is a process of integrating the signal response within each pixel area; artifacts in 3DGS are caused by the limited sampling bandwidth for the Gaussians that retrieves erroneous responses, especially when the pixel footprint changes drastically [2, 37]. It is possible to increase sampling bandwidth (*i.e.* via super sampling) or use prefiltering techniques to alleviate this problem; for example, Mip-Splatting [37] employs the prefiltering technique and presents a hybrid filtering mechanism to regularize the high-frequency components of 2D and 3D Gaussians to achieve anti-aliasing. While Mip-Splatting overcomes most aliasing in 3DGS, it is limited in capturing details and synthesizes over-smoothed results. Consequently, solving the integral of Gaussian signals within the pixel window area as intensity responses is crucial for both anti-aliasing and capturing details.

In this paper, we revisit pixel shading in 3DGS and introduce an analytic approximation of the window integral response of Gaussian signals for anti-aliasing. Rather than discrete sampling in 3DGS and prefiltering in Mip-Splatting, we analytically approximate the integral within each pixel area as shown in Fig. 1b. We term our method as *Analytic-Splatting*. Compared with Mip-Splatting, which approximates the pixel window as a 2D Gaussian low-pass filter, our proposed method does not suppress the high-frequency components in Gaussian signals and can better preserve high-quality details. Experiments show that our method removes the aliasing existing in 3DGS and other methods while synthesizing more details with better fidelity. We summarize our contributions as follows.

- We revisit the causes of aliasing in 3D Gaussian Splatting from the perspective of signal window response and derive an analytic approximation of the window response for Gaussian signals;
- Based on the derivation, we present *Analytic-Splatting* that improves the pixel shading in 3D Gaussian Splatting to achieve anti-aliasing and better detail fidelity, achieving state-of-the-art results on challenging datasets.

2 Related Works

Neural Rendering. Recently, neural rendering techniques exemplified by Neural Radiance Field (NeRF) [25] have achieved impressive results in novel view synthesis, and further enhanced several advanced tasks [15, 16, 23, 24, 27, 31, 35]. Nevertheless, the ray-tracing volume rendering used in NeRF hinders the real-time rendering performance, restricting the application prospects of NeRF. While several NeRF variants adopt efficient sampling strategies [21, 26, 36] or use explicit/hybrid representations [6, 9, 10, 30] with higher capacities, their sampling schemes are still costly and difficult to achieve real-time rendering. To overcome these limitations, 3DGS [18] employs forward mapping volume rendering technology and implements GPU-friendly tile-based rasterization to achieve real-time rendering and impressive rendering results. Due to its real-time rendering capability and impressive rendering performance, 3DGS has been widely used in advanced tasks such as Human/Avatar modeling [13, 28, 39, 41], surface reconstruction [7,11], inverse rendering [17,22,29], physical simulation [8,33], etc. Although rasterization makes 3DGS avoid tough sampling problems along rays and achieve promising results, it also introduces aliasing caused by restricted sampling bandwidth when shading pixels using 2D Gaussians. And the aliasing will be noticeable when the pixel footprint changes drastically (*e.g.* zooming in and out). In this paper, we study the errors introduced by the discrete sampling scheme used in 3DGS and introduce our advanced solution.

Anti-Aliasing. Aliasing is the phenomenon of overlapping frequency components when the discrete sampling rate is below the Nyquist rate. Anti-aliasing is critical for rendering high-fidelity results, which has been extensively explored in the computer graphics and vision community [1, 20, 32]. In the context of NeRFs, MipNeRF [2] and Zip-NeRF [4] pioneer the use of prefiltering and multi-sampling to address the aliasing issue in neural radiance fields (NeRF). Recent works also explored the anti-aliased NeRF for unbounded scenes [3], efficient reconstruction [14], and surface reconstruction [40]. All these works are built upon the ray-tracing volume rendering to consider the pixel footprint, by replacing the original ray-casting with cone-casting. However, the ray-tracing volume rendering is too computationally expensive to achieve real-time rendering. On the other hand, 3DGS [18] introduced real-time rasterization-based volume rendering but suffers from aliasing artifacts due to the discrete sampling during shading pixels using projected Gaussians. To this end, Mip-Splatting [37] presents a hybrid filtering mechanism to restrict the high-frequency components of 2D and 3D Gaussians to achieve anti-aliasing. Nevertheless, this low-pass filtering strategy hinders the capability to preserve high-quality details. In contrast, our approach introduces an analytic approximation of the integral within the pixel area to better capture the intensity response of each pixel, harvesting both aliasing-free and detail-preserving rendering results.

3 Preliminaries

In this section, we give the technical background necessary for the presentation of our proposed method.

3D Gaussian Splatting (3DGS) [18] explicitly represents 3D scene as a set of primitives $\{\boldsymbol{p}_i\}_{i=1}^N$. Given any primitive $\boldsymbol{p} \in \{\boldsymbol{p}_i\}_{i=1}^N$, 3DGS models it as a 3D Gaussian signal with mean vector $\boldsymbol{\mu} \in \mathbb{R}^3$ (*i.e.* the position of primitive $\boldsymbol{p}$) and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$. Note that $\boldsymbol{\Sigma}$ is factorized into a scaling matrix $\boldsymbol{S}$ and a rotation matrix $\boldsymbol{R}$ as $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top \boldsymbol{R}^\top$.

Given an extrinsic matrix $\boldsymbol{T}$ and a projection matrix $\boldsymbol{K}$, we get the projected position $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$ in 2D screen space as:

$$\hat{\boldsymbol{\mu}} = \boldsymbol{K}\boldsymbol{T}[\boldsymbol{\mu}, 1]^\top, \quad \hat{\boldsymbol{\Sigma}} = \boldsymbol{J}\boldsymbol{T}\boldsymbol{\Sigma}\boldsymbol{T}^\top \boldsymbol{J}^\top, \tag{1}$$

where $\boldsymbol{J}$ is the Jacobian matrix of the affine approximation of the perspective projection [42–44]. Note that 3DGS only retains the second-order values of $\hat{\boldsymbol{\mu}}$

and $\hat{\boldsymbol{\Sigma}}$ as $\hat{\boldsymbol{\mu}} \in \mathbb{R}^2$ and $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{2\times 2}$ respectively. The projected 2D Gaussian signal for the pixel $\boldsymbol{u}$ is given:

$$g^{\text{2D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) = \exp\left(-\frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{\mu}})^\top \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{u}-\hat{\boldsymbol{\mu}})\right), \tag{2}$$

Using the projected 2D Gaussian signal, 3DGS derives the volume transmittance and shades the color of pixel $\boldsymbol{u}$ through:

$$\boldsymbol{C}(\boldsymbol{u}) = \sum_{i=1}^{N} T_i g_i^{\text{2D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i)\alpha_i \boldsymbol{c}_i, \quad T_i = \prod_{j=1}^{i-1}(1 - g_j^{\text{2D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)\alpha_j), \tag{3}$$

where the symbols with the subscript i indicate the attributes related to the primitive $\boldsymbol{p}_i$. Specifically, α_i and $\boldsymbol{c}_i$ respectively denote the opacity and view-dependent color of primitive $\boldsymbol{p}_i$.

For better understanding, we further formulate the 2D Gaussian signal g^{2D} in Eq. (2) as a flattened expression. Considering $\hat{\boldsymbol{\Sigma}}$ is a real-symmetric 2×2 matrix, we numerically express $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\Sigma}}^{-1}$ as:

$$\begin{gathered} \hat{\boldsymbol{\Sigma}} = \begin{bmatrix} \hat{\Sigma}_{11} & \hat{\Sigma}_{12} \\ \hat{\Sigma}_{12} & \hat{\Sigma}_{22} \end{bmatrix}, \quad \hat{\boldsymbol{\Sigma}}^{-1} = \frac{1}{|\hat{\boldsymbol{\Sigma}}|}\begin{bmatrix} \hat{\Sigma}_{22} & -\hat{\Sigma}_{12} \\ -\hat{\Sigma}_{12} & \hat{\Sigma}_{11} \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \\ a = \frac{\hat{\Sigma}_{22}}{|\hat{\boldsymbol{\Sigma}}|}, \quad b = -\frac{\hat{\Sigma}_{12}}{|\hat{\boldsymbol{\Sigma}}|}, \quad c = \frac{\hat{\Sigma}_{11}}{|\hat{\boldsymbol{\Sigma}}|}, \end{gathered} \tag{4}$$

given the pixel $\boldsymbol{u} = [u_x, u_y]^\top$ and the projected position $\hat{\boldsymbol{\mu}} = [\hat{\mu}_x, \hat{\mu}_y]^\top$ in 2D screen space, Eq. (2) can be rewrited as:

$$\begin{gathered} g^{\text{2D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) = \exp\left(-\frac{a}{2}\hat{u}_x^2 - \frac{c}{2}\hat{u}_y^2 - b\hat{u}_x\hat{u}_y\right), \\ \hat{u}_x = u_x - \hat{\mu}_x, \quad \hat{u}_y = u_y - \hat{\mu}_y. \end{gathered} \tag{5}$$

Remark. It is worth noting that 3DGS treats each pixel as an isolated, single point when calculating its corresponding Gaussian value, as shown in Eq. (5). This approximating scheme functions effectively when training and testing images to capture the scene content from a relatively consistent distance. However, when the pixel footprint changes due to focal length or camera distance adjustments, 3DGS renderings exhibit considerable artifacts, such as thin Gaussians observed during zooming in. As a result, it is crucial to define the pixel as a window area and calculate the corresponding intensity response by integrating the Gaussian signal within this domain. Rather than using the intuitive but time-consuming super sampling, it would be better to tackle the problem more analytically, given that the Gaussian signal is a continuous function.

4 Methods

In Sec. 3, we observe that 3DGS ignores the window area of each pixel, and only considers the Gaussian value corresponding to the pixel center as its intensity

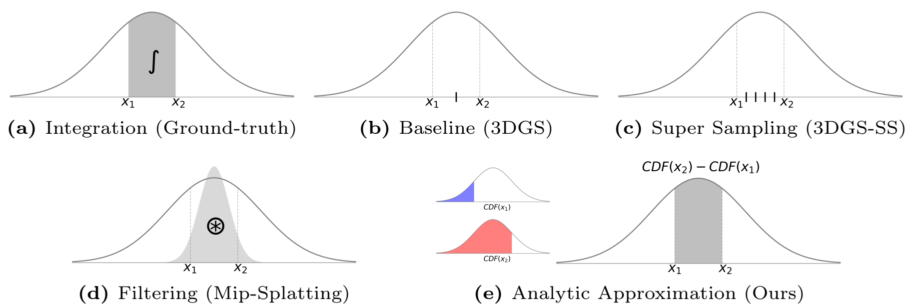


Fig. 2: Example diagram of the signal integration within window area and the approximation schemes used in different methods.

response. This approach would inevitably produce artifacts due to fluctuations of the pixel footprint under different resolutions. To address this problem, we are motivated to derive an analytical approximation of a 2D Gaussian signal within the pixel window area to accurately describe the intensity response of the imaging pixel. Subsequently, we plan to apply this derived integration to replace g^{2D} in the 3DGS framework.

4.1 Revisiting One-dimensional Gaussian Signal Response

Before describing our proposed method, we first revisit the example of the integrated response of a one-dimensional Gaussian signal within a window area for better understanding. Given a signal $g(x)$ and a window area $[x_1, x_2]$, we aim to get the response by integrating the signal $\mathcal{I}_g = \int_{x_1}^{x_2} g(x)\mathrm{d}x$ within this domain as shown in Fig. 2a. For an unknown signal, Monte Carlo sampling within the window area is a feasible approach to approximate the integral as demonstrated in Fig. 2b and Fig. 2c, and the approximation result $\mathcal{I}_g \approx \frac{x_2 - x_1}{N} \sum_{i=1}^{N} g(x_i),\ x_i \in [x_1, x_2]$ will be more accurate as the number of samples N increases. Nonetheless, increasing the number of samples (i.e., super-sampling) increases the computational burden significantly.

Fortunately, our goal in the 3DGS framework is to obtain the intensity response of the Gaussian signal within the window area $[x_1, x_2]$. Given that the Gaussian signal is a continuous real-valued function, it is natural to derive an analytical approximation to the Gaussian definite integral (Fig. 2a) which is more accurate compared to the numerical integration (Fig. 2b and Fig. 2c). For instance, in Mip-Splatting, the window area is treated as a Gaussian kernel g_{w}, and the integral is approximated as the result of sampling after convolving the Gaussian signal with the Gaussian kernel $\mathcal{I}_g \approx g \circledast g_{\mathrm{w}}$[1] as shown in Fig. 2d. While this prefiltering approximates the integral well in most cases, it still intro-

[1] the standard deviation σ in g_{w} is set to 0.1. Note that the result of convolving a Gaussian signal with a Gaussian kernel is still a Gaussian signal.

duces a large gap when the Gaussian signal g mainly consists of high-frequency components (*i.e.*, with small standard deviation σ).

To overcome these shortcomings, we are motivated to calculate the integral within the window area analytically. Specifically, the problem of evaluating the definite integral within $[x_1, x_2]$ can be simplified to the subtraction of two improper integrals by applying the first part of the fundamental theorem of calculus. Let $G(x)$ be the cumulative distribution function (CDF) of the standard Gaussian distribution $g(x)$ defined by,

$$G(x) = \int_{-\infty}^{x} g(u)\mathrm{d}u, \quad g(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right), \tag{6}$$

and the definite integral of $g(x)$ within $[x_1, x_2]$ can be expressed as,

$$\mathcal{I}_g = G(x_2) - G(x_1). \tag{7}$$

However, this CDF of the Gaussian function (defined using the error function erf) is not closed-form. We start by approximating the CDF $G(x)$ and find an important corollary, *i.e.*,

Definition 1. *The logistic function $S(x)$ is an analytic approximation of the CDF $G(x)$ with standard deviation $\sigma = 1$, which is defined as [5],*

$$S(x) = \frac{1}{1 + \exp\left(-1.6 \cdot x - 0.07 \cdot x^3\right)}, \tag{8}$$

This analytic approximation also shares properties of the CDF $G(x)$:

1. $S(x)$ is *non-decreasing* and *right-continuous*, satisfying

$$\lim_{x \to -\infty} G(x) = \lim_{x \to -\infty} S(x) = 0 \quad \text{and} \quad \lim_{x \to +\infty} G(x) = \lim_{x \to +\infty} S(x) = 1.$$

2. The curve of $S(x)$ has *2-fold rotational symmetry* around the point $(0, 1/2)$,

$$G(x) + G(-x) = S(x) + S(-x) = 1, \ \forall x \in \mathbb{R}.$$

For any Gaussian signals with different standard deviations, we can approximate their CDFs by scaling x in Eq. (8) by the reciprocals of their standard deviations. Once x in $S(x)$ scales by the reciprocal of σ, we express it as $S_\sigma(x)$. For more details, please refer to the *supplementary material*.

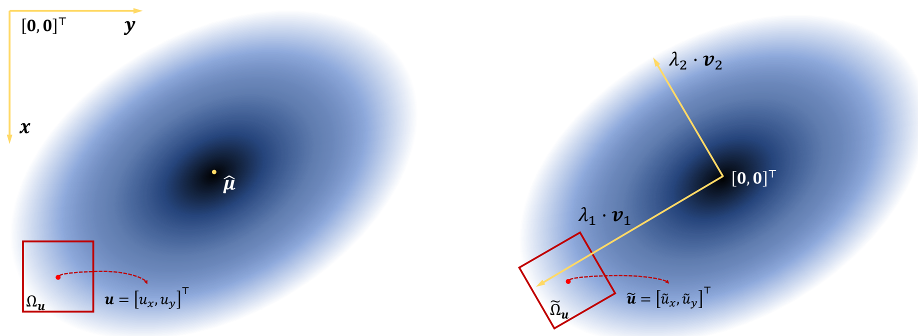
In summary, given the sample u and setting the window area as 1, the integral $\mathcal{I}_g(u)$ of Gaussian signal $g(x)$ within the area $[u - \frac{1}{2}, u + \frac{1}{2}]$ is defined as:

$$\mathcal{I}_g(u) = \int_{u-\frac{1}{2}}^{u+\frac{1}{2}} g(x)\mathrm{d}x = G(u + \frac{1}{2}) - G(u - \frac{1}{2}). \tag{9}$$

Now, according to Definition 1, we apply Eq. (8) and approximate the integral of Eq. (9) as:

$$\mathcal{I}_g(u) \approx S(u + \frac{1}{2}) - S(u - \frac{1}{2}). \tag{10}$$

In the following section, we will discuss how to employ the above approximation to 2D-pixel shading in Analytic-Splatting.



(a) Integral over the original pixel area. **(b)** Integral over the rotated pixel area.

Fig. 3: Example diagram of the pixel integration domain in Eq. (11) and the domain after rotation in Eq. (13). The yellow lines in Fig. 3a are the coordinate axes of 2D screen space; And the yellow lines in Fig. 3b are the eigenvectors scaled by the eigenvalues.

4.2 Analytic-Splatting

After revisiting the one-dimensional Gaussian signal integration, we proceed with applying the analytic approximation to the integral of projected 2D Gaussians over pixel window areas, which we call *Analytic-Splatting*. Mathematically, we replace the sampled $g^{2\mathrm{D}}(\boldsymbol{u})$ in Eq. (3) with the approximated integral $\mathcal{I}_g^{2\mathrm{D}}(\boldsymbol{u})$. For the pixel $\boldsymbol{u} = [u_x, u_y]^\top$ in 2D screen space, which corresponds to the window area $\Omega_{\boldsymbol{u}}$ as shown in Fig. 3a, the integration of Gaussian signal in Eq. (2) is represented as:

$$\mathcal{I}_g^{2\mathrm{D}}(\boldsymbol{u}) = \int_{u_x-\frac{1}{2}}^{u_x+\frac{1}{2}} \int_{u_y-\frac{1}{2}}^{u_y+\frac{1}{2}} e^{\left(-\frac{a}{2}(x-\hat{\mu}_x)^2 - \frac{c}{2}(y-\hat{\mu}_y)^2 - \underbrace{b(x-\hat{\mu}_x)(y-\hat{\mu}_y)}_{\text{correlation term}}\right)} \mathrm{d}x\mathrm{d}y. \tag{11}$$

Notably, handling the correlation term in this integral is intractable. To unravel the correlation term and thus feasibly solve the integral, we diagonalize the covariance matrix $\hat{\boldsymbol{\Sigma}}$ of the 2D Gaussian $g^{2\mathrm{D}}$ and slightly rotate the integration area as shown in Fig. 3b, thus approximating the integral by multiplying two independent 1D Gaussian integrals.

In detail, we first perform eigenvalue decomposition on the covariance matrix $\hat{\boldsymbol{\Sigma}}$ (refer to Eq. (4)) to obtain eigenvalues $\{\lambda_1, \lambda_2\}$ and the corresponding eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2\}$. After diagonalization, for better description, we take $\hat{\boldsymbol{\mu}} = [\hat{\mu}_x, \hat{\mu}_y]^\top$ (the mean vector of $g^{2\mathrm{D}}$) as the origin and the eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2\}$ as the axis to construct a new coordinate system. In this coordinate system, given a pixel $\boldsymbol{u} = [u_x, u_v]^\top$, we rewrite the $g^{2\mathrm{D}}$ in Eq. (5) as the multiplication of two independent 1D Gaussians:

$$
\begin{aligned}
g^{2\mathrm{D}}(\boldsymbol{u}) &= \exp\left(-\frac{\tilde{u}_x^2}{2\lambda_1}-\frac{\tilde{u}_y^2}{2\lambda_2}\right) = \exp\left(-\frac{\tilde{u}_x^2}{2\lambda_1}\right)\exp\left(-\frac{\tilde{u}_y^2}{2\lambda_2}\right), \\
\tilde{\boldsymbol{u}} &= \begin{bmatrix}\tilde{u}_x\\ \tilde{u}_y\end{bmatrix} = \begin{bmatrix}- \boldsymbol{v}_1 -\\ - \boldsymbol{v}_2 -\end{bmatrix}(\boldsymbol{u}-\hat{\boldsymbol{\mu}}) = \begin{bmatrix}- \boldsymbol{v}_1 -\\ - \boldsymbol{v}_2 -\end{bmatrix}\begin{bmatrix}u_x-\hat{\mu}_x\\ u_y-\hat{\mu}_y\end{bmatrix},
\end{aligned} \tag{12}
$$

where $\tilde{\boldsymbol{u}} = [\tilde{u}_x, \tilde{u}_y]^\top$ denotes the diagonalized coordinate of the pixel center. After diagonalization, we further rotate the pixel integration area $\Omega_{\boldsymbol{u}}$ along the pixel center to align it with the eigenvectors and get $\tilde{\Omega}_{\boldsymbol{u}}$ for approximating the integral. Thus the integral in Eq. (11) can be approximated as:

$$
\begin{aligned}
\mathcal{I}_g^{2\mathrm{D}}(\boldsymbol{u}) &\approx \int_{\tilde{\Omega}_{\boldsymbol{u}}} g^{2\mathrm{D}}(\boldsymbol{u})\mathrm{d}\boldsymbol{u} = \int_{\tilde{u}_x-\frac{1}{2}}^{\tilde{u}_x+\frac{1}{2}} e^{\left(-\frac{x^2}{2\lambda_1}\right)}\mathrm{d}x \int_{\tilde{u}_y-\frac{1}{2}}^{\tilde{u}_y+\frac{1}{2}} e^{\left(-\frac{y^2}{2\lambda_2}\right)}\mathrm{d}y \\
&= 2\pi\sqrt{\lambda_1\lambda_2}\int_{\tilde{u}_x-\frac{1}{2}}^{\tilde{u}_x+\frac{1}{2}} \frac{1}{\sqrt{2\pi\lambda_1}} e^{\left(-\frac{x^2}{2\lambda_1}\right)}\mathrm{d}x \int_{\tilde{u}_y-\frac{1}{2}}^{\tilde{u}_y+\frac{1}{2}} \frac{1}{\sqrt{2\pi\lambda_2}} e^{\left(-\frac{y^2}{2\lambda_2}\right)}\mathrm{d}y \\
&\approx 2\pi\sigma_1\sigma_2\left[S_{\sigma_1}(\tilde{u}_x+\frac{1}{2})-S_{\sigma_1}(\tilde{u}_x-\frac{1}{2})\right]\left[S_{\sigma_2}(\tilde{u}_y+\frac{1}{2})-S_{\sigma_2}(\tilde{u}_y-\frac{1}{2})\right],
\end{aligned} \tag{13}
$$

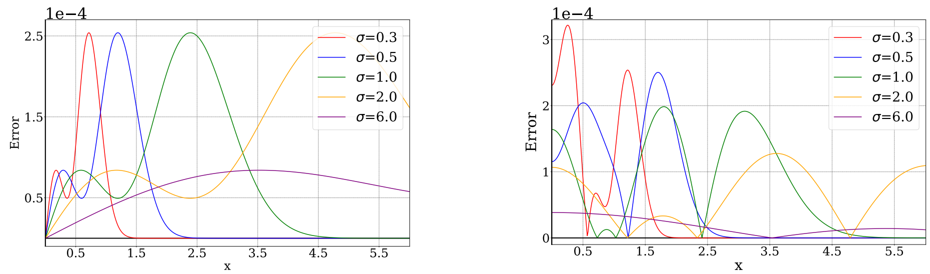
where σ_* subscripts of G and S respectively correspond to Gaussian signals with the standard deviation σ_*. $\sigma_1 = \sqrt{\lambda_1}$ and $\sigma_2 = \sqrt{\lambda_2}$ denote the standard derivations of the independent Gaussian signals along two eigenvectors respectively. In summary, the volume shading in Analytic-Splatting is given by:

$$
\begin{aligned}
\boldsymbol{C}(\boldsymbol{u}) &= \sum_{i=1}^{N} T_i \mathcal{I}_{g-i}^{2\mathrm{D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i)\alpha_i \boldsymbol{c}_i, \quad T_i = \prod_{j=1}^{i-1}(1-\mathcal{I}_{g-j}^{2\mathrm{D}}(\boldsymbol{u}|\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)\alpha_j), \\
\mathcal{I}_g^{2\mathrm{D}}(\boldsymbol{u}) &= 2\pi\sigma_1\sigma_2\left[S_{\sigma_1}(\tilde{u}_x+\frac{1}{2})-S_{\sigma_1}(\tilde{u}_x-\frac{1}{2})\right]\left[S_{\sigma_2}(\tilde{u}_y+\frac{1}{2})-S_{\sigma_2}(\tilde{u}_y-\frac{1}{2})\right].
\end{aligned} \tag{14}
$$

5 Experiments

5.1 Approximation Error Analysis

In this section, we first comprehensively explore the approximation errors in our scheme and then conduct an elaborate comparison against other schemes. It is worth noting that during training, the pruning and densification schemes proposed in 3DGS tend to maintain the standard deviations of Gaussian signals within an appropriate range (*i.e.* $\sigma \in [0.3,\ 6.6]$), and each Gaussian signal only responds to pixels within the 99% confidence interval (*i.e.* $|x| < 3\sigma$) for shading. Therefore, we only consider the Gaussian signals with standard deviations in $[0.3,\ 6.6]$, and merely discuss the approximation error of pixels within the 99% confidence interval.



(a) Approximation errors of our conditioned logistic function Eq. (8) to the CDF Eq. (6) referring to different standard deviations.

(b) Approximation errors of our approximated window integral Eq. (10) to the ground-truth Eq. (9) referring to different standard deviations.

Fig. 4: Error Analysis of using our conditioned logistic function Eq. (8) to approximate CDF of Gaussian signals Eq. (6) and window integration Eq. (9). Note that the scaling factor of the error is $1e^{-4}$.

Referring to Eq. (6) and Eq. (8), we get the approximation error function about the CDF of the Gaussian function:

$$\mathcal{E}_{\mathrm{CDF}}(x) = \left| S(x) - G(x) \right|, \tag{15}$$

and referring to Eq. (9) and Eq. (10), the approximation error function regarding the 1-width window area integral response is:

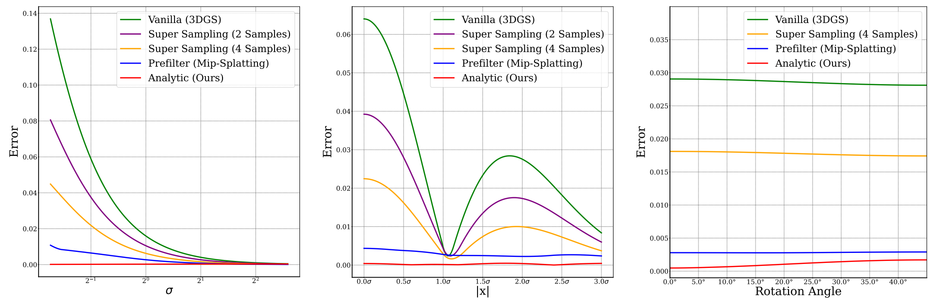
$$\mathcal{E}_{\mathrm{Int}}(x) = \left| \left(S(x+\frac{1}{2}) - S(x+\frac{1}{2}) \right) - \left(G(x+\frac{1}{2}) - G(x+\frac{1}{2}) \right) \right|, \tag{16}$$

the approximation error $\mathcal{E}_{\mathrm{CDF}}$ and $\mathcal{E}_{\mathrm{Int}}$ referring to different standard deviations are shown in Fig. 4a and Fig. 4b respectively.[2] Please note that the errors shown in Fig. 4 are scaled by a tiny factor of 1e−4.

For approximating the integral response over the 1-width window area, our scheme significantly outperforms other schemes. Fig. 5a and Fig. 5b respectively show that one-dimensional approximation error with different standard deviations and values. Our scheme is robust to these two conditions, especially when the standard deviations of Gaussian signals become smaller, our advantage becomes more obvious, which means that our scheme is better at capturing the high-frequency signals (*i.e.* details) of the scene, and our subsequent experimental results also verify this.

Last but not least, we employ this scheme to approximate the window integral responses of two-dimensional Gaussian signals, which requires rotating the integration domain from $\Omega_{\boldsymbol{u}}$ to $\tilde{\Omega}_{\boldsymbol{u}}$ as shown in Fig. 3 and inevitably introduces additional errors. To study this error, we record the approximation errors caused

[2] Since Eq. (15) and Eq. (16) are even functions, we show the results for the positive semi-axis over $0 \leq x \leq 6$ in Fig. 4.



(a) Approximation errors referring to different standard deviations.

(b) Approximation errors for different values.

(c) Approximation errors of rotating the integral domain by different angles in Fig. 3.

Fig. 5: Error Analysis of approximating the window integral Eq. (9) using different schemes in Fig. 2.

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.
NeRF w/o $\mathcal{L}_{\text{area}}$	31.20	30.65	26.25	22.53	27.66	0.950	0.956	0.930	0.871	0.927	0.055	0.034	0.043	0.075	0.052
NeRF [25]	29.90	32.13	33.40	29.47	31.23	0.938	0.959	0.973	0.962	0.958	0.074	0.040	0.024	0.039	0.044
MipNeRF [2]	32.63	34.34	35.47	35.60	34.51	0.958	0.970	0.979	0.983	0.973	0.047	0.026	0.017	0.012	0.026
Plenoxels [9]	31.60	32.85	30.26	26.63	30.34	0.956	0.967	0.961	0.936	0.955	0.052	0.032	0.045	0.077	0.051
TensoRF [6]	32.11	33.03	30.45	26.80	30.60	0.956	0.966	0.962	0.939	0.956	0.056	0.038	0.047	0.076	0.054
Instant-NGP [26]	30.00	32.15	33.31	29.35	31.20	0.939	0.961	0.974	0.963	0.959	0.079	0.043	0.026	0.040	0.047
Tri-MipRF [14]	32.65	34.24	35.02	35.53	34.36	0.958	0.971	0.980	0.987	0.974	0.047	0.027	0.018	0.012	0.026
3DGS [18]	28.79	30.66	31.64	27.98	29.77	0.943	0.962	0.972	0.960	0.960	0.065	0.038	0.025	0.031	0.040
3DGS-SS [18]	32.05	33.78	33.92	31.12	32.71	0.964	0.975	0.980	0.977	0.974	0.039	0.021	0.016	0.020	0.024
Mip-Splatting [37]	32.81	34.49	35.45	35.50	34.56	0.967	0.977	0.983	0.988	0.979	0.035	0.019	0.013	0.010	0.019
Ours	33.22	34.92	35.98	36.00	35.03	0.967	0.977	0.984	0.989	0.979	0.033	0.019	0.012	0.010	0.018

Table 1: Quantitative Comparison of Analytic-Splatting against several cutting-edge methods on the multi-scale Blender Synthetic dataset [2]. These methods conduct multi-scale training and testing.

by rotating the integration domain from 0° to 45°[3] under different standard deviations of Gaussian signals and distributions, as shown in Fig. 5c. Although the approximation error of our scheme slightly increases as the rotation angle becomes larger, our scheme still surpasses other schemes.

5.2 Comparison

To verify the anti-aliasing capability and versatility of Analytic-Splatting, we conduct experiments against state-of-the-art methods under the multi-scale training & multi-scale testing (MTMT) setting on Blender Synthetic [2,25] and Mip-NeRF 360 [3] datasets. We further evaluate our methods against cutting-edge methods on several scene datasets [3,12,19] under the widely used single-scale training & single-scale testing (STST) setting.

Dataset & Metric. For the MTMT setting, we conduct experiments using benchmark datasets of multi-scale Blender Synthetic [2,25] and multi-scale Mip-NeRF 360 [3]. They respectively contain 8 objects and 9 scenes, each object and

[3] Since we always hold $\sigma_1 \geq \sigma_2$ in practice, thus we only consider the approximation error caused by the rotation angle from 0° to 45°.

Fig. 6: Qualitative comparison of full-resolution and low-resolution (1/8) on Multi-Scale Blender [2]. All methods are trained on images with downsampling rates covering [1, 2, 4, 8]. Our method can better overcome the artifacts in 3DGS with better fidelity of details.

scene is compiled by downscaling the original dataset with a factor of 2, 4, and 8, and combining. For the Blender Synthetic dataset, each object contains 100 images for training and 200 images for testing. For the Mip-NeRF 360 dataset, we select 1 image from every 8 images for testing and the remaining images for training. For the STST setting, we follow 3DGS [18] and evaluate three real scene datasets [3, 12, 19]. To verify the efficacy of our method, we evaluate the synthesized novel view on these datasets in terms of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [38].

Implementation. We implement Analytic-Splatting upon 3DGS [18] and customize our shading module with CUDA extensions. Following 3DGS, we train Analytic-Splatting using the same parameters, training schedule, and loss functions, ensuring the efficacy of our scheme. To achieve super sampling of Gaussian signals, we implement 3DGS-SS, which first renders an image at twice the target

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.
Mip-NeRF 360 [3]	27.50	29.19	30.45	30.86	29.50	0.778	0.864	0.912	0.931	0.871	0.254	0.136	0.077	0.058	0.131
Zip-NeRF [4]	28.25	30.01	31.56	32.52	30.58	0.822	0.891	0.933	0.955	0.900	0.198	0.099	0.056	0.038	0.098
3DGS [18]	26.55	28.00	28.51	27.45	27.63	0.779	0.854	0.891	0.888	0.853	0.274	0.162	0.102	0.087	0.156
3DGS-SS [18]	27.20	28.75	29.89	29.71	28.89	0.800	0.871	0.914	0.928	0.878	0.246	0.138	0.081	0.061	0.131
Mip-Splatting [37]	27.20	28.74	29.90	30.66	29.12	0.802	0.870	0.915	0.944	0.883	0.244	0.146	0.090	0.056	0.134
Ours	27.50	28.99	30.35	31.21	29.51	0.808	0.874	0.919	0.945	0.887	0.231	0.132	0.077	0.051	0.123

Table 2: Quantitative Comparison of Analytic-Splatting against several cutting-edge methods on the multi-scale Mip-NeRF 360 dataset [3, 4]. These methods conduct multi-scale training and testing.

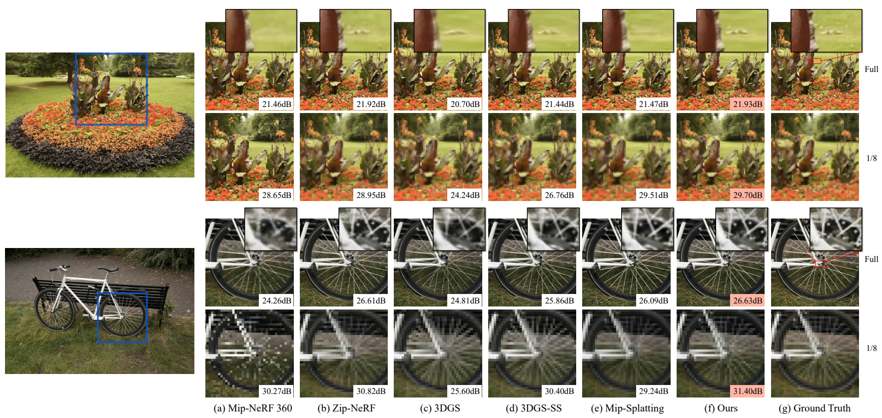


Fig. 7: Qualitative comparisons of full-resolution and low-resolution on Multi-Scale Mip-NeRF 360 [3, 4]. All methods are trained on images with downsampling rates covering [1, 2, 4, 8]. Our method can better overcome the artifacts with better fidelity of details. Please note that the artifacts of 3DGS/3DGS-SS become obvious at low resolutions, especially on elongated shapes (*e.g.* wheels and flower stems), while Mip-Splatting produces over-smoothed results (*e.g.* lobes).

resolution and obtains the final image at the target resolution through average pooling. For the MTMT setting, we follow previous works [2, 14, 37] and tend to select more full-resolution images as supervision samples during training. Please refer to the *supplementary material* for more details on the backpropagation implementation of rendering.

Evaluation on Blender Synthetic Dataset. We compare our method with several state-of-the-art methods *i.e.* NeRF [25], MipNeRF [2], Plenoxels [9], TensoRF [6], Instant-NGP [26], Tri-MipRF [14], 3DGS [18] and its variants (*i.e.* 3DGS-SS, Mip-Splatting [37]) on the Blender Synthetic dataset. The quantitative results in Tab. 1 show that Analytic-Splatting outperforms other methods in all aspects. The qualitative results in Fig. 6 demonstrate that Analytic-Splatting can better capture high-frequency details while being anti-aliased. More results can be found in the *supplementary material*.

Evaluation on Mip-NeRF 360 Dataset. We compare our method with several cutting-edge methods *i.e.* Mip-NeRF 360 [3], Zip-NeRF [4], 3DGS [18] and its variants (*i.e.* 3DGS-SS, Mip-Splatting [37]) on the challenging Mip-NeRF 360 dataset. The results of Zip-NeRF are reported from the available official implementation[4] . Please note that Mip-NeRF 360 and Zip-NeRF struggle with real-time rendering, especially Zip-NeRF performs super sampling techniques in the rendering phase. The quantitative results in Tab. 2 show that our method is second only to Zip-NeRF, and we outperform other methods with real-time rendering capability (*i.e.* 3DGS and its variants). The qualitative comparisons in Fig. 7 demonstrate that our method has better anti-aliasing capability and detail fidelity despite facing complex scenes. More results can be found in the *supplementary material*.

Single-Scale Evaluation on Scene Datasets We evaluate our method against other methods on complex scene datasets (*i.e.* Mip-NeRF 360 [3], Tanks & Temples [19], and Deep Blending [12]) under the single-scale training and testing setting, which is the most widely used setting. The overall results are shown in Tab. 3, our method shows great generalization across different datasets and almost outperforms other methods.

	Mip-NeRF 360			Tanks&Temples			Deep Blending		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Plenoxels [9]	23.08	0.625	0.463	21.07	0.721	0.379	23.06	0.795	0.510
INGP-Base [26]	25.30	0.671	0.371	21.72	0.734	0.330	23.62	0.797	0.423
INGP-Big [26]	25.59	0.699	0.331	21.92	0.752	0.305	24.96	0.817	0.390
Mip-NeRF 360 [3]	27.69	0.792	0.237	22.22	0.800	0.257	29.40	0.901	0.245
3DGS [18]	27.21	0.815	0.214	23.14	0.844	0.183	29.41	0.903	0.243
Mip-Splatting [37]	27.57	0.817	0.218	23.78	0.851	0.178	29.69	0.904	0.248
Ours	27.58	0.816	0.217	23.84	0.851	0.177	29.75	0.905	0.248

Table 3: Quantitative comparison of our method against previous methods over three datasets [3, 12, 19]. All methods are trained on full-resolution images and tested on the same-resolution images.

6 Conclusion

In this paper, we first revisit the window response of 1D Gaussian signals and reason about an analytical and accurate approximation using a conditioned logistic function. We then apply this approximation in the 2D-pixel shading and present *Analytic-Splatting*, which approximates the pixel area integral response to achieve anti-aliasing capability and better detail fidelity. Our extensive experiments demonstrate the efficacy of Analytic-Splatting in achieving state-of-the-art novel view synthesis results under the multi-scale setting.

Limitations. Compared with 3DGS and Mip-Splatting, our shading implementation introduces more root and exponential operations, which inevitably increases the computational burden and reduces the frame rate. Despite this, our frame rate is only 10% lower than Mip-Splatting in a few cases.

[4] https://github.com/jonbarron/camp_zipnerf

Acknowledgements

This work is supported by the Guangdong Provincial Key Laboratory of Human Digital Twin (2022B1212010004).

References

1. Akeley, K.: Reality engine graphics. In: Conference on Computer graphics and interactive techniques (1993)
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2023)
5. Bowling, S.R., Khasawneh, M.T., Kaewkuekool, S., Cho, B.R.: A logistic approximation to the cumulative normal distribution. Journal of industrial engineering and management **2**(1), 114–127 (2009)
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
7. Chen, H., Li, C., Lee, G.H.: Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. arXiv preprint arXiv:2312.00846 (2023)
8. Feng, Y., Feng, X., Shang, Y., Jiang, Y., Yu, C., Zong, Z., Shao, T., Wu, H., Zhou, K., Jiang, C., et al.: Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. arXiv preprint arXiv:2401.15318 (2024)
9. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022)
10. Gao, Q., Xu, Q., Su, H., Neumann, U., Xu, Z.: Strivec: Sparse tri-vector radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17569–17579 (2023)
11. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. arXiv preprint arXiv:2311.12775 (2023)
12. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG) **37**(6), 1–15 (2018)
13. Hu, L., Zhang, H., Zhang, Y., Zhou, B., Liu, B., Zhang, S., Nie, L.: Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. arXiv preprint arXiv:2312.02134 (2023)
14. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19774–19783 (2023)

15. Huang, X., Zhang, Q., Feng, Y., Li, H., Wang, X., Wang, Q.: Hdr-nerf: High dynamic range neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18398–18408 (2022)
16. Huang, Z., Liang, Z., Zhang, H., Lin, Y., Jia, K.: Sur2f: A hybrid representation for high-quality and efficient surface reconstruction from multi-view images. arXiv preprint arXiv:2401.03704 (2024)
17. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. arXiv preprint arXiv:2311.17977 (2023)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
19. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
20. Kuznetsov, A.: Neumip: Multi-resolution neural materials. ACM Transactions on Graphics (ToG) **40**(4) (2021)
21. Li, R., Tancik, M., Kanazawa, A.: Nerfacc: A general nerf acceleration toolbox. arXiv preprint arXiv:2210.04847 (2022)
22. Liang, Z., Zhang, Q., Feng, Y., Shan, Y., Jia, K.: Gs-ir: 3d gaussian splatting for inverse rendering. arXiv preprint arXiv:2311.16473 (2023)
23. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5741–5751 (2021)
24. Ma, L., Li, X., Liao, J., Zhang, Q., Wang, X., Wang, J., Sander, P.V.: Deblur-nerf: Neural radiance fields from blurry images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12861–12870 (2022)
25. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision. pp. 405–421 (2020)
26. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
27. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
28. Saito, S., Schwartz, G., Simon, T., Li, J., Nam, G.: Relightable gaussian codec avatars. arXiv preprint arXiv:2312.03704 (2023)
29. Shi, Y., Wu, Y., Wu, C., Liu, X., Zhao, C., Feng, H., Liu, J., Zhang, L., Zhang, J., Zhou, B., et al.: Gir: 3d gaussian inverse rendering for relightable scene factorization. arXiv preprint arXiv:2312.05133 (2023)
30. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
31. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
32. Wu, L., Zhao, S., Yan, L.Q., Ramamoorthi, R.: Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. ACM Transactions on Graphics (ToG) **38**(4), 1–14 (2019)

33. Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y., Jiang, C.: Physgaussian: Physics-integrated 3d gaussians for generative dynamics. arXiv preprint arXiv:2311.12198 (2023)
34. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
35. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems **34**, 4805–4815 (2021)
36. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
37. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. arXiv preprint arXiv:2311.16493 (2023)
38. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
39. Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. arXiv preprint arXiv:2312.02155 (2023)
40. Zhuang, Y., Zhang, Q., Feng, Y., Zhu, H., Yao, Y., Li, X., Cao, Y.P., Shan, Y., Cao, X.: Anti-aliased neural implicit surfaces with encoding level of detail. In: SIGGRAPH Asia 2023 Conference Papers. pp. 1–10 (2023)
41. Zielonka, W., Bagautdinov, T., Saito, S., Zollhöfer, M., Thies, J., Romero, J.: Drivable 3d gaussian avatars. arXiv preprint arXiv:2311.08581 (2023)
42. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa volume splatting. In: Proceedings Visualization, 2001. VIS'01. pp. 29–538. IEEE (2001)
43. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 371–378 (2001)
44. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa splatting. IEEE Transactions on Visualization and Computer Graphics **8**(3), 223–238 (2002)