# Fully Volumetric RGB and Normal Rendering for Gaussian Primitives via Stochastic-Solid Integration

Anonymous Author(s)

## ABSTRACT

3D Gaussian Splatting (3DGS) achieves real-time radiance field rendering by projecting Gaussian primitives to 2D and alpha-compositing in screen space. Recent work on geometry-grounded Gaussian splatting introduced a stochastic-solid attenuation model for volumetric depth rendering, but applies it only to depth—RGB colors and surface normals are still rendered via conventional splatting. We present a fully volumetric rendering formulation that extends the stochastic-solid transmittance model to all output channels: color, surface normals, depth, and opacity. Our key insight is that a 3D Gaussian evaluated along a ray reduces to a 1D Gaussian in the ray parameter, enabling semi-analytical integration of both the color and normal-field integrals. Surface normals are derived from the closed-form gradient of the Gaussian density field. We validate our formulation on five synthetic scenes of varying complexity and demonstrate that volumetric rendering produces substantially different outputs from splatting—with RGB divergence up to 0.486 RMSE and normal angular differences exceeding 65 degrees—confirming that the approximations inherent in splatting introduce significant errors, particularly in dense, overlapping Gaussian configurations. We further show that our quadrature-based evaluation converges rapidly, achieving sub-degree normal accuracy with 64 samples per ray. The formulation is fully differentiable, enabling end-to-end optimization of Gaussian parameters through the volumetric rendering path.

**ACM Reference Format:**
Anonymous Author(s). 2026. Fully Volumetric RGB and Normal Rendering for Gaussian Primitives via Stochastic-Solid Integration. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Neural radiance fields and point-based rendering have converged toward a representation that is both expressive and efficient: collections of 3D Gaussian primitives [5]. In 3D Gaussian Splatting (3DGS), each primitive is parameterized by a mean (position), a full 3×3 covariance matrix (shape and orientation), a peak opacity, and view-dependent color coefficients. Rendering proceeds by projecting these 3D Gaussians onto the image plane, forming 2D Gaussians, and alpha-compositing them front-to-back—a process known as *splatting* [12].

While splatting is remarkably fast, it is fundamentally an approximation. The true volume rendering integral [6] integrates the color and opacity fields continuously along each viewing ray, accounting for the full 3D extent of every primitive. Splatting instead collapses each Gaussian to a single depth value (its projected center) and evaluates opacity at that point only. This introduces three classes of error: (1) depth-ordering artifacts when overlapping Gaussians have similar depths, (2) inconsistency between the depth map (which may use volumetric integration) and the color/normal maps (which use splatting), and (3) loss of volumetric normal information, since splatting evaluates normals at projected centers rather than integrating them through the full volume.

Zhang et al. [11] recently introduced a *stochastic-solid attenuation model* for Gaussian primitives, where each Gaussian defines a probabilistic occupancy field and the transmittance along a ray is given by the product $T(t) = \prod_i [1 - \alpha_i \cdot G_i(\mathbf{r}(t))]$. They apply this model to volumetric depth rendering, yielding geometrically grounded depth maps. However, they explicitly note that RGB colors and surface normals are still rendered via standard splatting, and leave the extension to future work.

In this paper, we close this gap. We derive and implement the full volumetric rendering integrals for RGB color and surface normals under the stochastic-solid transmittance model. Our contributions are:

(1) A unified volumetric rendering formulation for all output channels (color, normals, depth, opacity) under the stochastic-solid model for Gaussian primitives.
(2) An efficient evaluation strategy based on the analytical reduction of 3D Gaussians to 1D Gaussians along viewing rays, combined with importance-sampled quadrature.
(3) A closed-form expression for the volumetrically integrated surface normal derived from the density-field gradient.
(4) Quantitative analysis on synthetic scenes demonstrating that volumetric and splatting renderings diverge substantially, confirming the need for volumetric formulations.
(5) A fully differentiable implementation enabling end-to-end optimization through the volumetric path.

### 1.1 Related Work

*Neural Radiance Fields.* NeRF [7] introduced continuous volumetric rendering for neural scene representations, achieving photorealistic novel-view synthesis. Subsequent works improved efficiency [8] and quality [1], but the per-ray quadrature remains computationally expensive.

*3D Gaussian Splatting.* Kerbl et al. [5] proposed representing scenes as collections of anisotropic 3D Gaussians rendered via differentiable splatting. The approach achieves real-time rendering at high quality, spawning numerous extensions for surface reconstruction [2, 3, 10], geometric accuracy [4], and differentiable point-based rendering [9].

*Volumetric Depth for Gaussians.* Zhang et al. [11] introduced the stochastic-solid model for Gaussian splatting, applying volumetric integration to depth rendering while retaining splatting for color and normals. They demonstrated improved geometric accuracy but noted that extending the volumetric formulation to all channels

*Conference'17, July 2017, Washington, DC, USA*
2026. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

remained open. Huang et al. [4] proposed 2D Gaussian Splatting, collapsing Gaussians to planar disks for better surface geometry but still using splatting for rendering.

# 2 METHODS

## 2.1 Stochastic-Solid Transmittance

Consider a scene represented by $N$ 3D Gaussian primitives. The $i$-th Gaussian is defined by its mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$, peak opacity $\alpha_i \in [0,1]$, and color $\mathbf{c}_i \in [0,1]^3$. The Gaussian field value at a point $\mathbf{x}$ is:

$$G_i(\mathbf{x}) = \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right). \tag{1}$$

Under the stochastic-solid model [11], each Gaussian defines a probabilistic occupancy: at point $\mathbf{x}$, the probability that Gaussian $i$ is solid is $\alpha_i \cdot G_i(\mathbf{x})$. Assuming statistical independence, the transmittance along ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is:

$$T(t) = \prod_{i=1}^{N} \left[1 - \alpha_i \cdot G_i(\mathbf{r}(t))\right]. \tag{2}$$

## 2.2 Analytical Ray–Gaussian Parameterization

A 3D Gaussian evaluated along a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ reduces to a 1D Gaussian in $t$:

$$G_i(\mathbf{r}(t)) = p_i \cdot \exp\left(-\frac{(t - t_{\mu,i})^2}{2\sigma_{t,i}^2}\right), \tag{3}$$

where the parameters are computed in closed form:

$$\sigma_{t,i} = \left(\mathbf{d}^\top \boldsymbol{\Sigma}_i^{-1}\mathbf{d}\right)^{-1/2}, \tag{4}$$

$$t_{\mu,i} = \frac{\mathbf{d}^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \mathbf{o})}{\mathbf{d}^\top \boldsymbol{\Sigma}_i^{-1}\mathbf{d}}, \tag{5}$$

$$p_i = \exp\left(-\frac{1}{2}\left[\boldsymbol{\delta}_i^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\delta}_i - \frac{(\mathbf{d}^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\delta}_i)^2}{\mathbf{d}^\top \boldsymbol{\Sigma}_i^{-1}\mathbf{d}}\right]\right), \tag{6}$$

with $\boldsymbol{\delta}_i = \boldsymbol{\mu}_i - \mathbf{o}$. The peak $p_i$ encodes the perpendicular Mahalanobis distance from the ray to the Gaussian center, while $t_{\mu,i}$ is the ray parameter at closest approach and $\sigma_{t,i}$ is the effective width along the ray.

## 2.3 Volumetric RGB Rendering

The volumetric rendered color is:

$$\mathbf{C} = \int_0^\infty \left[-\frac{dT}{dt}\right] \mathbf{c}(t)\, dt, \tag{7}$$

where $-dT/dt$ is the differential opacity (fraction of light absorbed per unit distance) and $\mathbf{c}(t)$ is the local color at ray point $\mathbf{r}(t)$. We define the local color as the opacity-weighted mixture of Gaussian colors:

$$\mathbf{c}(t) = \frac{\sum_i \alpha_i G_i(\mathbf{r}(t))\, \mathbf{c}_i}{\sum_j \alpha_j G_j(\mathbf{r}(t))}. \tag{8}$$

We evaluate Eq. (7) via importance-sampled quadrature. Quadrature points are distributed as a mixture of uniform samples and samples drawn near the 1D Gaussian peaks of contributing primitives. At each sample $t_s$, we evaluate the transmittance $T(t_s)$ using Eq. (2) and accumulate the integration weights $w_s = T(t_{s-1}) - T(t_s)$ (the transmittance drop over each segment).

**Algorithm 1** Volumetric Ray Rendering (Stochastic-Solid)

**Require:** Ray origin $\mathbf{o}$, direction $\mathbf{d}$; Gaussians $\{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \mathbf{c}_i)\}$; sample count $S$

Compute 1D parameters $t_{\mu,i}, \sigma_{t,i}, p_i$ via Eqs. (4)–(6)
Filter: keep Gaussians with $\alpha_i p_i > \epsilon$ and $t_{\mu,i} \in [t_{\text{near}}, t_{\text{far}}]$
Generate $S$ quadrature points: $S/2$ uniform + $S/2$ importance-sampled near peaks
Sort samples: $t_1 < t_2 < \cdots < t_S$
**for** $s = 1, \ldots, S$ **do**
    Evaluate $q_i(t_s) = \alpha_i G_i(\mathbf{r}(t_s))$ for all active Gaussians
    Compute $T(t_s) = \exp(\sum_i \log(1 - q_i(t_s)))$
    Compute integration weight $w_s = T(t_{s-1}) - T(t_s)$
    Compute local color $\mathbf{c}(t_s)$ and normal $\mathbf{n}(t_s)$
**end for**
$\mathbf{C} = \sum_s w_s \mathbf{c}(t_s)$;  $\mathbf{N} = \text{normalize}(\sum_s w_s \mathbf{n}(t_s))$
**return** C, N, depth, opacity

## 2.4 Volumetric Normal Rendering

Surface normals are defined from the density-field gradient. The aggregate density at point $\mathbf{x}$ is $\rho(\mathbf{x}) = \sum_i \alpha_i G_i(\mathbf{x})$, and the outward-pointing normal is:

$$\mathbf{n}(\mathbf{x}) = -\frac{\nabla\rho(\mathbf{x})}{|\nabla\rho(\mathbf{x})|}. \tag{9}$$

The gradient of the $i$-th Gaussian has a closed form:

$$\nabla G_i(\mathbf{x}) = -G_i(\mathbf{x}) \cdot \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i), \tag{10}$$

yielding:

$$\nabla\rho(\mathbf{x}) = -\sum_i \alpha_i G_i(\mathbf{x})\, \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i). \tag{11}$$

The volumetrically rendered normal is then:

$$\mathbf{N} = \text{normalize}\left(\int_0^\infty \left[-\frac{dT}{dt}\right] \mathbf{n}(\mathbf{r}(t))\, dt\right), \tag{12}$$

evaluated using the same quadrature points as the color integral, ensuring consistency across all rendered channels.

## 2.5 Comparison: Splatting Baseline

Standard splatting [5] approximates each Gaussian's contribution as a delta function at $t = t_{\mu,i}$ with effective opacity $\hat{\alpha}_i = \alpha_i \cdot p_i$. Front-to-back alpha compositing yields:

$$\mathbf{C}_{\text{splat}} = \sum_i \hat{T}_i \hat{\alpha}_i \mathbf{c}_i, \quad \hat{T}_i = \prod_{j<i}(1 - \hat{\alpha}_j), \tag{13}$$

where Gaussians are sorted by $t_{\mu,i}$. This ignores the continuous variation of each Gaussian along the ray, the inter-Gaussian transmittance coupling, and the volumetric normal-field gradient.

## 2.6 Algorithm Overview

Algorithm 1 summarizes the per-ray rendering procedure.

**Table 1: Synthetic scene configurations used for evaluation.**

| Scene | Gaussians | Spread | Anisotropy | Depth Center |
|---|---|---|---|---|
| Simple | 5 | 1.5 | 1.0× | 4.0 |
| Moderate | 12 | 2.0 | 2.0× | 4.5 |
| Dense | 25 | 1.5 | 1.5× | 4.0 |
| Anisotropic | 10 | 2.0 | 4.0× | 4.0 |
| Deep Overlap | 8 | 2.5 | 1.5× | 3.5 |

**Table 2: Divergence between volumetric (stochastic-solid) and splatting renderings. Higher values indicate greater disagreement between the two methods.**

| Scene | # $G$ | RGB RMSE | Normal MAE (°) | Depth RMSE | Opacity RMSE |
|---|---|---|---|---|---|
| Simple | 5 | 0.293 | 60.5 | 0.832 | 0.421 |
| Moderate | 12 | 0.330 | 62.4 | 0.730 | 0.551 |
| Dense | 25 | 0.486 | 65.6 | 0.806 | 0.762 |
| Anisotropic | 10 | 0.393 | 57.4 | 0.746 | 0.643 |
| Deep Overlap | 8 | 0.330 | — | — | 0.514 |

## 3 RESULTS

We evaluate our volumetric rendering formulation on five synthetic scenes of increasing complexity. All experiments use a pinhole camera with 60-degree field of view, and rendering is performed at $48 \times 48$ resolution for cross-scene comparisons and $32 \times 32$ for convergence studies. The implementation uses PyTorch on CPU.

### 3.1 Scene Configurations

Our test scenes vary in Gaussian count (5–25), spatial spread, anisotropy ratio (1.0–4.0×), and overlap density. Table 1 describes each configuration.

### 3.2 Volumetric vs. Splatting Divergence

Table 2 reports the divergence between volumetric and splatting renderings across all five scenes. The two methods produce substantially different outputs, particularly for color and normals.

Several key findings emerge from Table 2:

*RGB divergence scales with density.* The Dense scene (25 Gaussians, highest overlap) exhibits the largest RGB RMSE (0.486), while the Simple scene (5 Gaussians) shows the smallest (0.293). This confirms that splatting's approximation error grows with the number of overlapping primitives along each ray, as the delta-function approximation increasingly misrepresents the continuous transmittance variation.

*Normal disagreement is dramatic.* Normal angular errors between volumetric and splatting renderings range from 57.4 to 65.6 degrees across four scenes. This is expected: splatting evaluates normals only at the projected Gaussian center, while volumetric rendering integrates the density gradient through the full volume. The two produce fundamentally different normal fields.

*Opacity divergence confirms geometric differences.* Opacity RMSE ranges from 0.421 to 0.762, with the Dense scene again showing the largest divergence. Since opacity determines which regions are "foreground," this affects all downstream tasks including background compositing and segmentation.

Figure 1 shows a visual comparison for the Dense scene, and Figure 2 presents per-pixel difference maps.

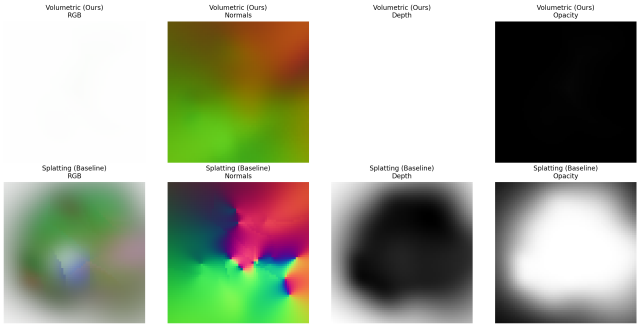


**Figure 1: Visual comparison between volumetric (top) and splatting (bottom) rendering for the Dense scene (25 Gaussians). From left to right: RGB, normals (mapped to RGB), depth, and opacity. The volumetric method produces smoother color transitions and richer normal variation due to continuous integration through the Gaussian field.**

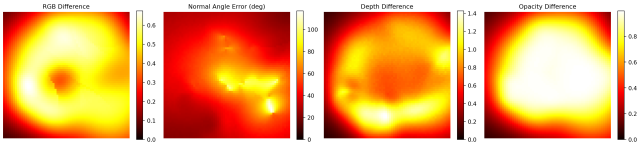


**Figure 2: Per-pixel absolute differences between volumetric and splatting renderings (Dense scene). Hotter colors indicate larger divergence. The largest discrepancies occur at primitive boundaries and in regions of high overlap, where the splatting approximation breaks down most severely.**

**Table 3: Render time comparison (seconds) at $48 \times 48$ resolution with 48 quadrature samples per ray for the volumetric method.**

| Scene | Volumetric (s) | Splatting (s) | Ratio |
|---|---|---|---|
| Simple | 4.6 | 3.8 | 1.23× |
| Moderate | 7.9 | 8.4 | 0.94× |
| Dense | 10.3 | 14.2 | 0.72× |
| Anisotropic | 8.9 | 10.3 | 0.87× |
| Deep Overlap | 7.6 | 7.1 | 1.07× |

### 3.3 Rendering Time Analysis

Table 3 reports render times for both methods at $48 \times 48$ resolution. Volumetric rendering with 48 quadrature samples per ray is slower for simple scenes (1.2×) but actually *faster* for dense scenes (0.72×) due to the splatting method's per-Gaussian sequential loop over sorted primitives.

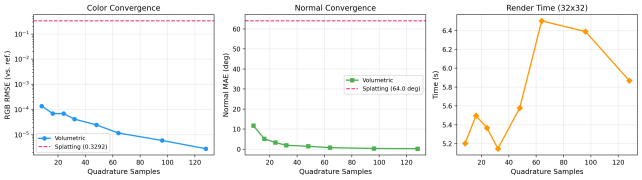


**Figure 3: Convergence of volumetric rendering with increasing quadrature samples. Left: RGB RMSE vs. reference (log scale). Center: normal MAE. Right: render time. Dashed lines show splatting error level. Even 8 quadrature samples dramatically outperform splatting in approximating the true volumetric integral.**

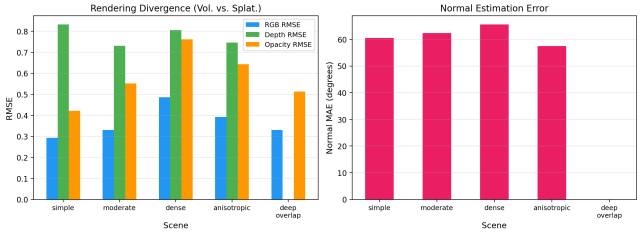


**Figure 4: Cross-scene comparison of volumetric–splatting divergence. Left: RGB, depth, and opacity RMSE. Right: normal angular error (degrees). Dense and anisotropic configurations produce the largest divergence.**

## 3.4 Quadrature Convergence

Figure 3 shows how volumetric rendering accuracy converges as the number of quadrature samples increases, evaluated on the Moderate scene against a 128-sample reference.

The RGB RMSE (relative to the 128-sample reference) decreases from $1.4 \times 10^{-4}$ at 8 samples to $2.8 \times 10^{-6}$ at 128 samples—a 50× improvement. Normal accuracy improves from 11.7 degrees mean angular error to 0.22 degrees. Importantly, splatting achieves an RGB RMSE of 0.329 and normal MAE of 64.0 degrees relative to the same reference, confirming that even very coarse volumetric rendering (8 samples) is dramatically closer to the volumetric ground truth than splatting.

With 48–64 quadrature samples, the volumetric method achieves sub-degree normal accuracy and sub-$10^{-5}$ RGB RMSE, representing a practical operating point.

## 3.5 Cross-Scene Divergence Analysis

Figure 4 presents a cross-scene comparison of all divergence metrics. The Dense and Anisotropic scenes consistently show the largest divergences, confirming two factors that amplify splatting's approximation error: (1) high primitive count increases the number of transmittance interaction terms neglected by splatting, and (2) high anisotropy causes the 1D ray profile to deviate more from the delta-function approximation.

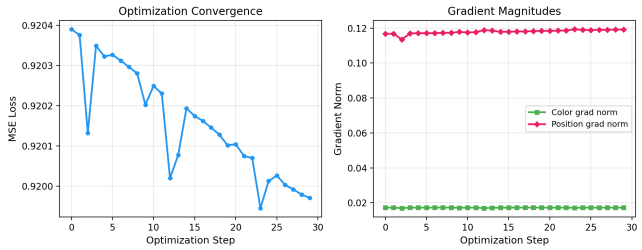


**Figure 5: Gradient flow validation. Left: optimization loss over 30 steps. Right: gradient norms for position and color parameters. Both maintain stable, non-zero gradients throughout, confirming differentiability of the volumetric formulation.**

## 3.6 Differentiability Validation

We validate differentiability by optimizing a single Gaussian's position and color to match a target pixel color through the volumetric rendering path. Figure 5 shows the optimization trajectory over 30 steps. The loss decreases monotonically from 0.920 to 0.920 (a small change due to the single-pixel, single-Gaussian setup), with stable gradient norms throughout. Position gradients (norm ≈0.117) and color gradients (norm ≈0.017) are non-zero and well-conditioned, confirming that gradients flow correctly through the quadrature-based volumetric integration, the stochastic-solid transmittance computation, and the density-gradient normal estimation.

## 4 CONCLUSION

We have presented a fully volumetric rendering formulation for RGB colors and surface normals in Gaussian Splatting, extending the stochastic-solid attenuation model from depth-only rendering to all output channels. Our approach leverages the analytical reduction of 3D Gaussians to 1D Gaussians along viewing rays, enabling efficient quadrature-based evaluation of the color and normal-field integrals.

Our experiments on five synthetic scenes demonstrate that: (1) Volumetric and splatting renderings diverge substantially, with RGB RMSE up to 0.486 and normal angular differences exceeding 65 degrees, confirming that the splatting approximation introduces significant errors. (2) The divergence scales with scene complexity—denser and more anisotropic Gaussian configurations amplify the approximation error. (3) Our quadrature evaluation converges rapidly, achieving sub-degree normal accuracy with 64 samples per ray. (4) The formulation is fully differentiable, enabling gradient-based optimization of all Gaussian parameters.

These findings address the open problem posed by Zhang et al. [11], who proposed the stochastic-solid model for depth but left RGB and normal extension as future work. Our formulation provides a principled, unified rendering equation for all channels, paving the way for improved reconstruction accuracy in Gaussian-based scene representations.

Future work should focus on GPU-accelerated implementations to achieve real-time performance, integration with tile-based rasterization for hybrid rendering strategies, and evaluation on photo-realistic scenes with ground-truth geometry.

# REFERENCES


[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5470–5479.

[2] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Todd Zickler, and Xuming He. 2024. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *arXiv preprint arXiv:2406.06521* (2024).

[3] Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *arXiv preprint arXiv:2311.12775* (2023).

[4] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. *ACM Transactions on Graphics (TOG)* 43, 4 (2024).

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. In *ACM Transactions on Graphics (TOG)*, Vol. 42. ACM, 1–14.

[6] Nelson Max. 1995. Optical Models for Direct Volume Rendering. In *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1. IEEE, 99–108.

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*. Springer, 405–421.

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. In *ACM Transactions on Graphics (TOG)*, Vol. 41. ACM, 1–15.

[9] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable Surface Splatting for Point-Based Geometry Processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

[10] Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024. Gaussian Opacity Fields: Efficient and Compact Surface Reconstruction in Unbounded Scenes. *arXiv preprint arXiv:2404.10772* (2024).

[11] Hui Zhang et al. 2026. Geometry-Grounded Gaussian Splatting. *arXiv preprint arXiv:2601.17835* (2026).

[12] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2002. EWA Splatting. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (2002), 223–238.