SPLAT THE NET: RADIANCE FIELDS WITH SPLATTABLE NEURAL PRIMITIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Radiance fields have emerged as a predominant representation for modeling 3D scene appearance. Neural formulations such as Neural Radiance Fields provide high expressivity but require costly ray marching for rendering, whereas primitive-based methods such as 3D Gaussian Splatting offer real-time efficiency through splatting, yet at the expense of representational power. Inspired by advances in both these directions, we introduce *splattable neural primitives*, a new volumetric representation that reconciles the expressivity of neural models with the efficiency of primitive-based splatting. Each primitive encodes a bounded neural density field parameterized by a shallow neural network. Our formulation admits an exact analytical solution for line integrals, enabling efficient computation of perspectively accurate splatting kernels. As a result, our representation supports integration along view rays without the need for costly ray marching. The primitives flexibly adapt to scene geometry and, being larger than prior analytic primitives, reduce the number required per scene. On novel-view synthesis benchmarks, our approach matches the quality and speed of 3D Gaussian Splatting while using $10\times$ fewer primitives and $6\times$ fewer parameters. These advantages arise directly from the representation itself, without reliance on complex control or adaptation frameworks.

1 INTRODUCTION

Radiance fields have become a predominant representation for modeling 3D scene appearance. Unlike surface-based approaches, their volumetric formulation is compatible with the gradient-based optimization routines employed during training from multi-view images. *Neural* representations in particular (Mildenhall et al., 2021) offer unprecedented expressivity in encoding radiance fields. However, rendering an image from a volumetric scene representation is generally challenging: Volume rendering (Kajiya & Von Herzen, 1984) requires the computation of costly integrals along view rays, typically solved using quadrature methods such as *ray marching* (Max, 1995). As a remedy, *primitive*-based representations have emerged as an efficient alternative. Popularized by 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023), these approaches model radiance fields using a mixture of simple volumetric functions. The key to high rendering efficiency lies in the observation that these primitives can be easily projected onto the image plane, where they become 2D kernels that can be efficiently *splatted*. A prime example is the 3D Gaussian primitive used in 3DGS, which reduces to a 2D Gaussian splatting kernel (Zwicker et al., 2001). Recently, a variety of functions have been explored as primitives (Mai et al., 2024; von Lützow & Nießner, 2025; Hamdi et al., 2024; Held et al., 2025a; Huang et al., 2024), all relying on relatively simple analytical formulations, which are widely considered essential for enabling efficient conversion into view-dependent splatting kernels.

These developments have led to a prevalent, somewhat dichotomous view of radiance field representations: Neural representations are expressive but come with the high cost of ray marching for rendering, whereas primitive-based representations, though simpler and less expressive, offer more efficient rendering through splatting. We challenge this common wisdom by introducing *splattable neural primitives*, offering both expressivity and real-time efficiency; see Fig. 1 for an overview.

The central design ingredient of our primitives is a neural volumetric density field. Its density distribution is parameterized by a shallow yet expressive neural network and is spatially bounded by an ellipsoid. This formulation admits an exact analytical solution for line integrals (Subr, 2021; Lloyd et al., 2020), which enables efficient computation of a perspectively accurate image-space footprint,

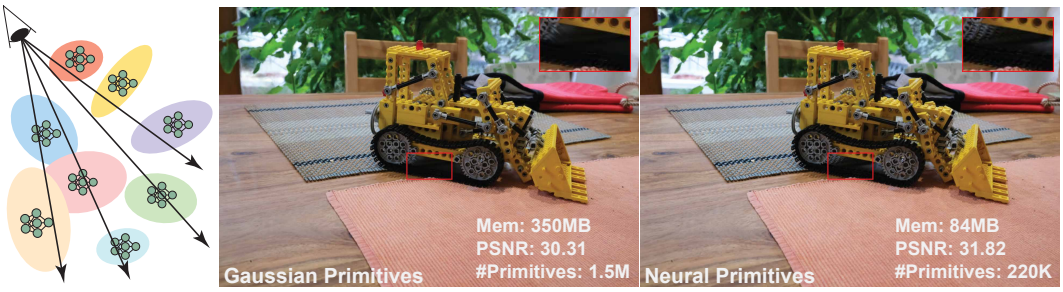

Figure 1: (a) Overview of *volumetric splattable neural primitives*. Each primitive is spatially bounded by an ellipsoid, and its density is parameterized as a shallow neural network. (b) A real scene rendered using Gaussian primitives (left) and neural primitives (right). Our method achieves comparable PSNR to the Gaussian representation but with fewer primitives, highlighting the expressivity of neural primitives.

i.e., a splatting kernel for rendering. Despite its neural representation, this enables integration of the density field along each pixel's view ray without the need for costly ray marching. Our primitives flexibly adapt to scene geometry and, being typically larger than the analytic primitives employed in recent work, reduce the total number needed to represent a scene. This yields a highly favorable trade-off among quality, performance, and compactness in novel-view synthesis: We match the quality and speed of 3DGS while using $10\times$ fewer primitives and $6\times$ fewer parameters. Crucially, these advantages result from the design of our representation itself, without requiring complex control or adaptation frameworks (Mallick et al., 2024; Fan et al., 2024). In summary, our contributions are:

- A taxonomy of radiance field representations, highlighting a dichotomy between neural and splatting-based approaches (Sec. 2.1).
- A novel volumetric representation based on splattable neural primitives, bridging the gap between and leveraging the benefits of both neural and primitive-based approaches (Sec. 3).
- The application of the representation to novel-view synthesis, validating its practical effectiveness and efficiency: Neural primitives achieve real-time rendering speed and produce result quality comparable to 3DGS with a smaller memory budget (Sec. 4).

We will make all source code and trained models available.

2 Background and Related Work

2.1 Radiance Field Representation and Rendering

Radiance fields represent the appearance of a 3D scene via a function $F_\theta : (\mathbf{x}, \mathbf{d}) \to (\sigma, \mathbf{c})$, which maps a spatial location $\mathbf{x} \in \mathbb{R}^3$ and view direction $\mathbf{d} \in \mathbb{S}^2$ to a volumetric density $\sigma \in \mathbb{R}_+$ and an RGB color $\mathbf{c} \in \mathbb{R}^3$. Synthesizing an image from a radiance field involves emission–absorption volume rendering (Kajiya & Von Herzen, 1984) along view rays $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ denotes the camera center, and $t \in \mathbb{R}_+$ parameterizes the ray. Specifically, each RGB pixel color $C \in \mathbb{R}^3$ is computed by evaluating the radiance field along its corresponding view ray:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} \exp\left(-\int_{t_n}^{t} \sigma\left(\mathbf{r}(s)\right) \mathrm{d}s\right) \sigma\left(\mathbf{r}(t)\right) \mathbf{c}\left(\mathbf{r}(t), \mathbf{d}\right) \mathrm{d}t, \tag{1}$$

where t_n and t_f are near and far bounds, respectively. Recent years have seen considerable research devoted to the foundational question of how to best represent F_θ. Representations can be arranged along an *atomicity scale*, from monolithic models that entangle all components in a single structure to modular formulations made up of many simple, spatially localized primitives (horizontal axis in Fig. 2a). In the following paragraphs, we briefly review the literature on radiance field representations along this atomicity scale, from monolithic to modular.

Neural Radiance Fields (NeRFs) (Mildenhall et al., 2021) represent F_θ using a neural network. Numerous follow-up works extended and enhanced its capabilities (e.g., (Barron et al., 2021; 2022; Martin-Brualla et al., 2021)). However, rendering an image from it involves ray marching, that is,

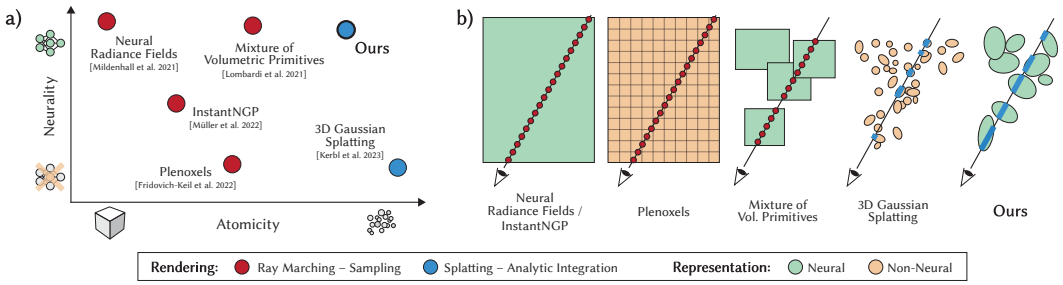

Figure 2: Positioning of our work relative to hallmark radiance field representations. *a)* Overview of representations organized along two central design dimensions: Atomicity (horizontal axis), spanning from monolithic (left) to distributed (right) representations; Neurality (vertical axis), ranging from non-neural (bottom) to fully neural (top) approaches. Dot color indicates the supported rendering algorithm. *b)* Illustration of the rendering algorithms associated with each representation. Our method is the only neural, primitive-based model that supports efficient splatting for rendering—thereby eliminating the need for costly ray marching—while retaining the flexibility of a neural design.

discretizing Eq. 1 into a finite sum of samples along each ray (Max, 1995) (red dots in Fig. 2b):

$$C(\mathbf{r}) \approx \sum_{i=1}^{N} \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \left(1 - \exp(-\sigma_i)\delta_i\right) \mathbf{c}_i, \tag{2}$$

where δ_i is the distance between adjacent samples. This rendering process is computationally expensive, as each sample entails a forward pass through the representation network. The pursuit of efficiency has motivated a shift from monolithic to more distributed, explicit representations. Prominent directions include the use of grids (Fridovich-Keil et al., 2022; Barron et al., 2023), volumetric meshes (Govindarajan et al., 2025), and neuro-explicit structures (Müller et al., 2022; Chan et al., 2022; Reiser et al., 2021), all seeking trade-offs between computational cost and memory. Sparse representations allow skipping of empty space (Lombardi et al., 2021; Liu et al., 2020; Yu et al., 2021), reducing the number of samples during ray marching.

Taking a radically different approach by pushing atomicity to the extreme, 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) represents F_θ as an unstructured mixture of up to millions of 3D primitive functions. Each primitive, P_i, is specified by a small set of parameters that determine its density distribution, $\sigma_i(\mathbf{x})$, along with appearance parameters that capture its view-dependent color, $\mathbf{c}_i(\mathbf{r})$. Rendering an image from this representation can be achieved through splatting, a two-step process: In the first step, σ_i is projected to the image plane by integrating along the view ray $\mathbf{r}$ (blue lines in Fig. 2b), yielding a 2D opacity kernel

$$\alpha_i(\mathbf{r}) = 1 - \exp\left(-\int_{-\infty}^{\infty} \sigma_i\left(\mathbf{r}(t)\right) \mathrm{d}t\right). \tag{3}$$

In the second step, Eq. 1 simplifies to highly efficient alpha blending of the 2D kernels:

$$C(\mathbf{r}) \approx \sum_{i \in \mathcal{N}(\mathbf{r})} \mathbf{c}_i(\mathbf{r}) \alpha_i(\mathbf{r}) \prod_{j=1}^{i-1} \left(1 - \alpha_j(\mathbf{r})\right), \tag{4}$$

where $\mathcal{N}$ represents the indices of the depth-sorted primitives intersected by the view ray. Evaluating Eq. 3 is generally non-trivial, but for 3D Gaussian primitives P_i, the footprint α_i simplifies to a 2D Gaussian kernel under reasonable assumptions (Zwicker et al., 2001; Celarek et al., 2025), enabling high rendering speed and supporting high-quality radiance fields with millions of Gaussians in 3DGS.

The 3D Gaussian is not the only primitive allowing efficient (approximate) conversion into a 2D splatting kernel α. Recent work has explored a variety of primitive shapes (Mai et al., 2024; von Lützow & Nießner, 2025; Hamdi et al., 2024; Held et al., 2025a; Chen et al., 2024; Li et al., 2024; Gu et al., 2024; Hamdi et al., 2024; Talegaonkar et al., 2025; Liu et al., 2025), all relying on hand-crafted *analytic* kernels for efficient evaluation of Eq. 3. In contrast, our representation introduces splattable *neural* primitives, greatly increasing modeling flexibility. Neural components have also been used in the context of primitive splatting, for example, by injecting structure into the representation (Lu et al., 2024) or enforcing spatial regularization (Mihajlovic et al., 2024), yet these methods ultimately splat

Gaussian functions. To the best of our knowledge, we are the first to represent the volumetric kernel itself – i.e., the density distribution – as a neural network, making the primitive *fundamentally neural* rather than merely Gaussian with neural augmentations.

2.2 Integration with Neural Networks

Estimating integrals is common in visual computing. Feed-forward neural networks trained on the integrand can sometimes perform this task effectively. A notable class in this context is shallow neural networks with one hidden layer, which remain universal function approximators (Cybenko, 1989) and can be integrated in closed form (Yan et al., 2013; Zhe-Zhao et al., 2006; Lloyd et al., 2020; Subr, 2021). Our approach builds on this insight by modeling neural primitives that support closed-form integration along view rays. In contrast, deep neural networks have also been applied to integral computation via derivative graphs (Lindell et al., 2021; Teichert et al., 2019), but their high evaluation cost and difficulty in producing consistent integrals along arbitrary rays remain challenges.

3 Method

In this section, we first introduce our neural representation (Sec. 3.1), before explaining how to render images using this representation (Sec. 3.2). Finally, we discuss implementation details including our population control strategy, network design, and training protocol (Sec. 3.3).

3.1 Representation

We parameterize F_θ as a mixture of volumetric primitives $\{P_i\}$. Each primitive occupies a volume bounded by an ellipsoid (Mai et al., 2024), which we denote as B (Fig. 3a). This ellipsoid is defined by a center $\mathbf{x}_B \in \mathbb{R}^3$, a scaling vector $\mathbf{s}_B \in \mathbb{R}^3$ along its principal axes, and a rotation quaternion $\mathbf{q}_B \in \mathbb{R}^4$. In accordance with the radiance field formalism, each primitive must define a spatially varying density σ and a view-dependent color $\mathbf{c}$, described next. We define a density field $\sigma(\mathbf{x}) : B \to \mathbb{R}$ within the volume of the ellipsoid as

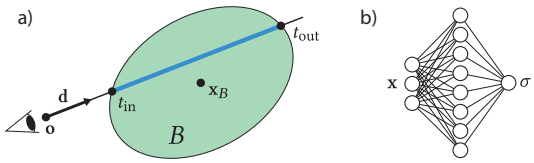

Figure 3: *a)* Geometry of our representation for a single primitive. Analytic splatting kernels are computed by performing closed-form integration of a neural density field (green shape) along view rays (blue line). *b)* Architecture of our neural density field. Density σ is a function of 3D spatial position $\mathbf{x}$.

$$\sigma(\mathbf{x}) = f_\sigma\left(\frac{\mathbf{x}-\mathbf{x}_B}{\|\mathbf{s}_B\|_\infty}\right), \tag{5}$$

where f_σ is a shallow neural network with one hidden layer of width N_σ and periodic activation (Sitzmann et al., 2020) (Fig. 3b):

$$f_\sigma(\mathbf{x}) = W_2\left(\cos\left(\omega_0\left(W_1\left(\mathbf{x}\right)+\mathbf{b}_1\right)\right)\right)+\mathbf{b}_2. \tag{6}$$

Here, $W_1 \in \mathbb{R}^{N_\sigma\times 3}$ and $W_2 \in \mathbb{R}^{1\times N_\sigma}$ are weight matrices, while $\mathbf{b}_1 \in \mathbb{R}^{N_\sigma}$ and $\mathbf{b}_2 \in \mathbb{R}$ are biases. Similar to (Sitzmann et al., 2020), we use a fixed boosting frequency ω_0, which yields a stable initialization. The network structure of Eq. 6 admits an interpretation analogous to a Fourier series, where W_1 and $\mathbf{b}_1$ correspond to frequencies and phases, and W_2 and $\mathbf{b}_2$ are amplitudes and offsets. The normalization by $\mathbf{x}_B$ and $\mathbf{s}_B$ in Eq. 5 ensures that f_σ operates on a centered and uniformly scaled domain. In the appendix, we show proof-of-concept extensions of this model to higher-dimensional inputs, including time, which can be easily incorporated into our model by augmenting the network's input dimensions. To represent view-dependent color, we adopt the Spherical Harmonics basis.

3.2 Rendering

Images of our representation are rendered using an efficient splatting-based approach. Specifically, for each primitive bounding ellipsoid B intersected by a view ray $\mathbf{r}(t)$, we compute the entry and exit distances, t_{in} and t_{out}, along the ray via an analytic line–ellipsoid intersection. To obtain a splatting kernel via Eq. 3, the crucial step is to evaluate the density integral along the view ray (blue line in Fig. 3a):

$$\hat{\alpha}(t_{\text{in}}, t_{\text{out}}, \mathbf{o}, \mathbf{d}) \coloneqq \int_{t_{\text{in}}}^{t_{\text{out}}} \sigma\left(\mathbf{o}+t\mathbf{d}\right)\mathrm{d}t = S\left(t_{\text{out}};\mathbf{o},\mathbf{d}\right) - S\left(t_{\text{in}};\mathbf{o},\mathbf{d}\right), \tag{7}$$

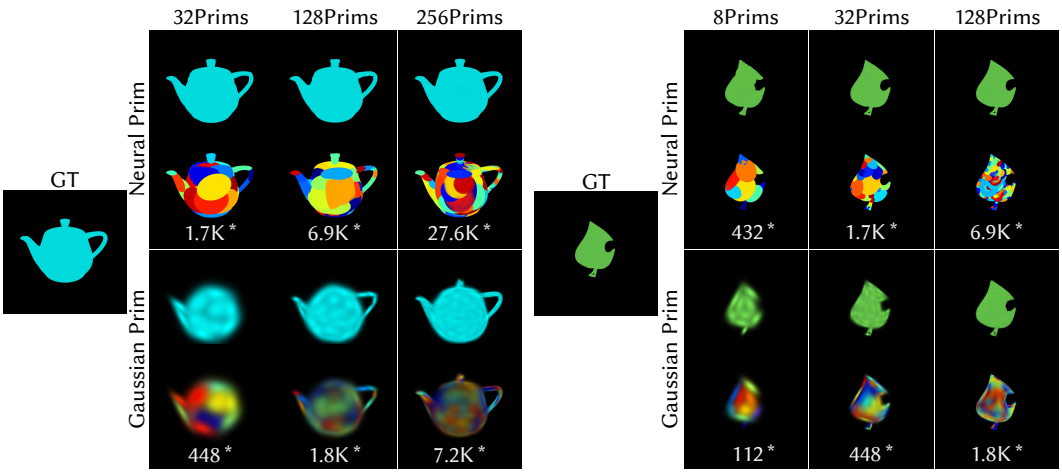

Figure 4: Demonstration of the expressivity of the proposed neural density field. We train both neural and Gaussian primitives on the *teapot* and *leaf* datasets using different numbers of primitives. For each example, we visualize the reconstructed density field and color-coded primitives, illustrating how these ellipsoid-bounded neural primitives are deformed to represent complex structures. $*$ denotes the total number of parameters.

where $S(t; \mathbf{o}, \mathbf{d})$ denotes the antiderivative with respect to t of the function $t \mapsto \sigma(\mathbf{o} + t\mathbf{d})$, which depends parametrically on $\mathbf{o}$ and $\mathbf{d}$. The equality follows from the fundamental theorem of calculus. Based on recent findings (Lloyd et al., 2020; Subr, 2021), we derive a closed-form antiderivative for our density field:

$$S(t; \mathbf{o}, \mathbf{d}) = \left[W_2 \oslash (\omega_0 \cdot W_1(\mathbf{d}))\right] \sin\left(\omega_0 \left(t \cdot W_1(\mathbf{d}) + W_1(\mathbf{o}) + \mathbf{b}_1\right)\right) + t \cdot \mathbf{b}_2, \tag{8}$$

where $\oslash$ denotes elementwise division. Incorporating Eq. 3 with the previous derivations yields the final splatting kernel

$$\alpha(\mathbf{r}) = 1 - \exp\left(-\max\left(0, \hat{\alpha}\left(t_{\text{in}}, t_{\text{out}}, \mathbf{o}, \mathbf{d}\right)\right)\right), \tag{9}$$

where the additional clamping to zero ensures nonnegative accumulated density. The final pixel color is determined using front-to-back compositing per Eq. 4.

Discussion We emphasize the efficiency of evaluating the splatting kernel via Eq. 9, which computes a closed-form integral along arbitrary view rays through the neural density field, thereby avoiding the computational cost of ray marching. In contrast to splatting-based Gaussian rendering, relying on an affine approximation of the projection operator (Heckbert, 1989; Zwicker et al., 2004), our method yields perspectively accurate results. Note that the density σ in Eq. 5 is never evaluated directly, neither during training nor during view synthesis. Instead, all computations operate directly on its antiderivative S. Yet, in contrast to a light-field-style approach that directly regresses integrated appearance (Sitzmann et al., 2021), our method achieves multi-view consistency by construction.

3.3 Implementation Details

Primitives We initialize $W_1 \sim \mathcal{U}(-1/3, 1/3)$ and $W_2 \sim \mathcal{U}\left(-\sqrt{6/N_\sigma}/\omega_0, \sqrt{6/N_\sigma}/\omega_0\right)$ following (Sitzmann et al., 2020). We set the number of hidden neurons N_σ to 8 and the frequency multiplier ω_0 to 30. Similar to 3DGS, we employ four bands of Spherical Harmonics coefficients for color representation. Each neural primitive in our system consists of 99 parameters in total, around $1.6\times$ more than Gaussian primitives used in 3DGS. We provide a detailed analysis of network configurations in Sec. 5.

Population Control Population control is a key factor to the success of primitive-based methods. However, the 3DGS densification strategy is incompatible with neural primitive representations. To address this, we introduce a simple yet effective densification strategy. Unlike 3DGS, which uses the gradients of primitive screen-space locations as the criterion for densification, our approach relies on the gradient magnitude of the network weights. Similar to 3DGS, we duplicate or split primitives when this gradient exceeds a threshold. Primitives with low gradients are pruned. We do not use any opacity resetting.

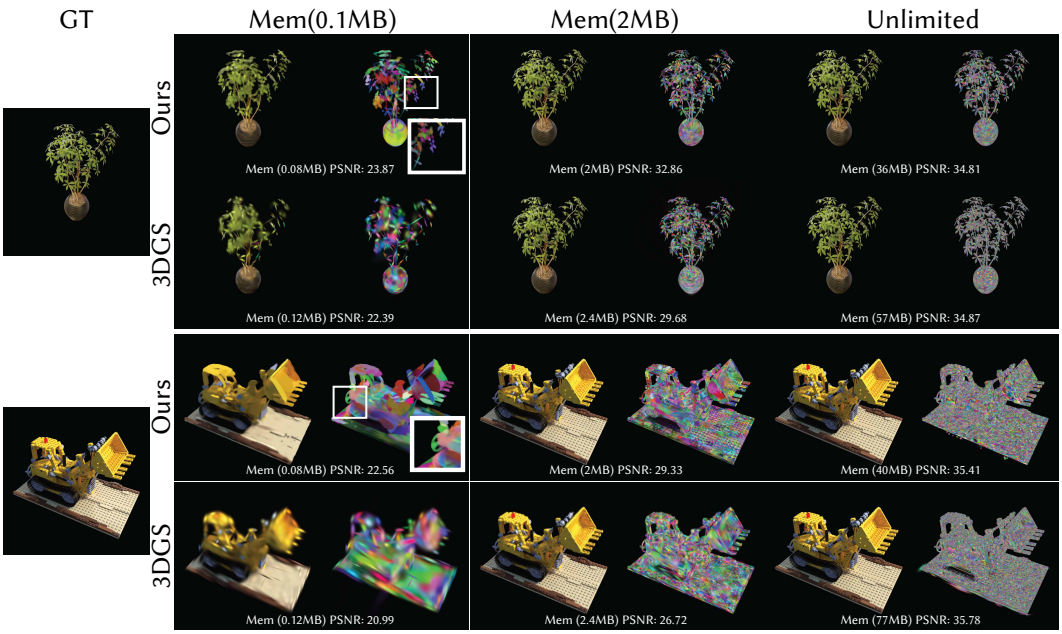

Figure 5: Comparison of our method against 3DGS on the synthetic dataset under different memory constraints.

Training We follow the same loss function as 3DGS, and introduce a geometric regularization term to penalize the extreme anisotropy in primitive shapes by minimizing the standard deviation of the components of the scale vector $\mathbf{s}_B$. The effectiveness of this regularization is demonstrated in Sec. 5. We implement all frameworks in PyTorch (Paszke et al., 2017) and CUDA. All models are trained on a single NVIDIA A40 GPU and evaluated on an NVIDIA RTX 4090 for performance analysis. Due to the complex optimization landscape of neural fields, the convergence of our representation is slower than a Gaussian-based one. We therefore extend training to $100k$ iterations. Additional training details are provided in Appendix A.

4 Evaluation

In this section, we perform a comprehensive evaluation of neural primitives on novel-view synthesis tasks. We first demonstrate the expressivity of the neural density field (Sec. 4.1). We then perform quantitative and qualitative analysis on synthetic datasets (Sec. 4.2) and real datasets (Sec. 4.3). Please refer to Appendix B for additional results.

4.1 Primitive Expressivity

Leveraging a flexible neural density field and analytically exact integration, our method faithfully reproduces complex geometries with a small number of primitives. To demonstrate this, we optimize varying numbers of neural and Gaussian primitives to approximate the density fields of several 3D geometries from multiple views. We visualize both the renderings and the color-coded primitives in Fig. 4. We observe that a few neural primitives suffice to represent complex and diverse geometries, such as the teapot's curved handle, the smooth cut in the leaf, and the triangular leaf petiole. In contrast, Gaussian primitives are limited by their symmetric ellipsoidal shape and soft boundaries, making them unsuitable for accurately representing complex solid structures. Neural primitives achieve superior performance while using $4\times$ fewer primitives and $16\times$ fewer parameters than Gaussian primitives.

4.2 Synthetic Scenes

Protocol We compare our method with 3DGS on the Synthetic NeRF dataset (Mildenhall et al., 2021) across varying memory budgets. Specifically, we resample the original meshes to target vertex counts (200, 500, 1k, 2k, 5k, 10k, 20k) and use them to initialize primitive positions for optimization,

Table 1: Quantitative comparison of our method against 3DGS on the Synthetic NeRF dataset under different memory budgets. We evaluate image quality using three standard metrics: LPIPS, PSNR, and SSIM.

Mem (MB)	0.1		0.4		1.0		2.0		4.0		Unlimited	
Method	3DGS	**Ours**	3DGS	**Ours**	3DGS	**Ours**	3DGS	**Ours**	3DGS	**Ours**	3DGS	**Ours**
PSNR↑	23.1	24.7	25.6	27.6	27.2	28.9	28.4	30.4	29.6	31.4	33.3	33.4
SSIM↑	.843	.879	.882	.916	.907	.932	.925	.948	.941	.956	.970	.967
LPIPS↓	.249	.161	.174	.097	.129	.073	.098	.051	.072	.039	.031	.032

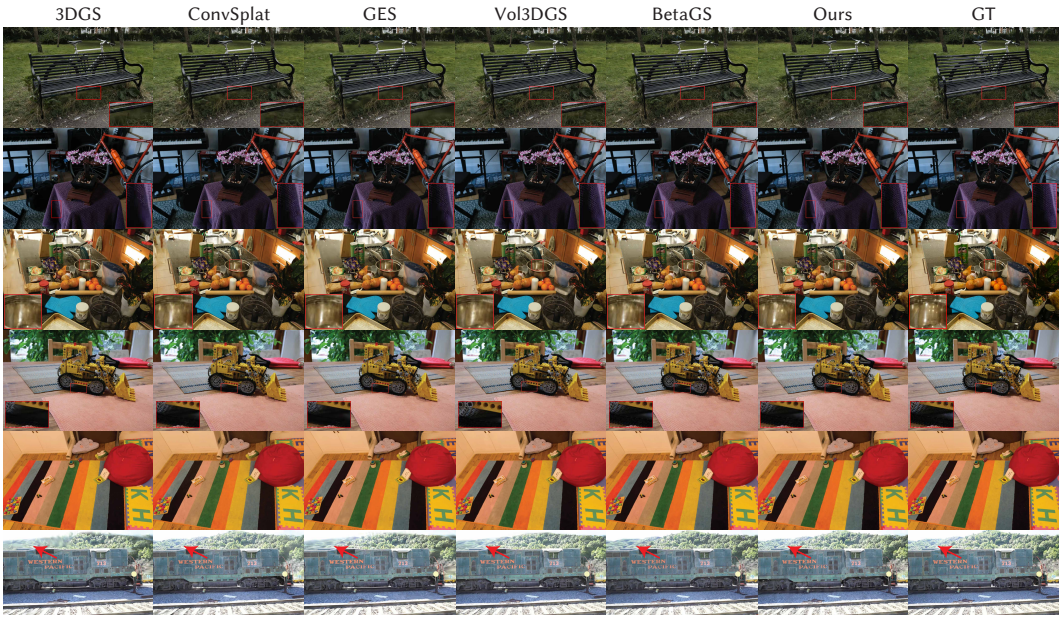

Figure 6: Visual comparison of our method against several primitive-based methods on the novel-view synthesis task for real scenes. We demonstrate that our neural primitives achieve high-fidelity results comparable to other approaches, requiring $10\times$ fewer primitives and $6\times$ fewer parameters.

omitting primitive densification. We also include an "unlimited" setting, in which training follows the standard densification procedure with no primitive budget.

Results We report numerical results in Tab. 1. Our method outperforms 3DGS under limited memory budgets and achieves performance comparable to 3DGS when no memory constraints are imposed. Visual comparisons are shown in Fig. 5. For *Ficus*, a single primitive can already reconstruct an entire leaf (highlighted by the white frame). Similarly, in *Lego*, neural primitives capture diverse geometries, such as the front shovel and the rear wheel. In contrast, Gaussian primitives perform poorly on these complex structures, particularly under tight budgets.

4.3 Real Scenes

Protocol For evaluation on real scenes, we follow established practice and use two scenes from Deep Blending (Hedman et al., 2018), two from Tanks & Temples (Knapitsch et al., 2017), and all scenes from the Mip-NeRF360 dataset (Barron et al., 2022). We compare against three method families: (i) splatting-based approaches with analytic primitives – 3DGS (Kerbl et al., 2023), GES (Hamdi et al., 2024), ConvSplat (Held et al., 2025b), BetaGS (Liu et al., 2025), and Vol3DGS (Talegaonkar et al., 2025); (ii) T-3DGS (Mallick et al., 2024), which provides a more sophisticated mechanism for controlling the memory footprint of primitive-based representations; and (iii) monolithic representations – Plenoxels (Fridovich-Keil et al., 2022), INGP (Müller et al., 2022), and MipNeRF360 (Barron et al., 2022). All experiments use the official code released by the respective authors. Since our reproduced baseline results closely match those reported in the respective papers, we report the original numbers for consistency. For a fair comparison, all inference FPS values are measured on a single NVIDIA GeForce RTX 4090 GPU.

Table 2: Numerical comparisons on three real-scene datasets. For each method, we indicate whether it is splatting-based (Spl.) and/or neural (Neu.). We also report novel-view synthesis quality (PSNR↑, SSIM↑ (Wang et al., 2004), LPIPS↓ (Zhang et al., 2018)), rendering speed (in frames per second), and memory usage (in MB).

			Mip-NeRF360					Tanks & Temples					Deep Blending				
	Spl.	Neu.	PSNR	SSIM	LPIPS	FPS	Mem	PSNR	SSIM	LPIPS	FPS	Mem	PSNR	SSIM	LPIPS	FPS	Mem
Plen	✗	✗	23.08	.626	.463	7	2.1k	21.08	.719	.379	13	2.3k	23.06	.795	.510	11	2.7k
INGP	✗	✓	25.59	.699	.331	9	48	21.92	.745	.305	14	48	24.96	.817	.390	3	48
Mip360	✗	✓	27.69	.792	.237	<1	9	22.22	.759	.257	<1	9	29.40	.901	.245	<1	9
3DGS	✓	✗	27.21	.815	.214	152	734	23.14	.841	.183	188	411	29.41	.903	.243	154	676
GES	✓	✗	26.91	.794	.250	279	377	23.35	.836	.198	372	222	29.68	.901	.252	289	399
BetaGS	✓	✗	28.75	.845	.179	71	356	24.85	.870	.140	119	200	30.12	.914	.236	91	343
ConvSplat	✓	✗	26.66	.769	.266	103	77	23.71	.842	.170	83	83	29.61	.901	.245	66	110
Vol3DGS	✓	✗	27.30	.813	.209	124	703	23.74	.854	.167	168	255	29.72	.908	.247	156	844
T-3DGS	✓	✗	27.31	.801	.252	265	152	23.95	.837	.201	408	73	29.82	.904	.260	409	67
Ours	✓	✓	27.21	.791	.216	115	93	23.59	.846	.162	158	80	29.20	.892	.264	178	82

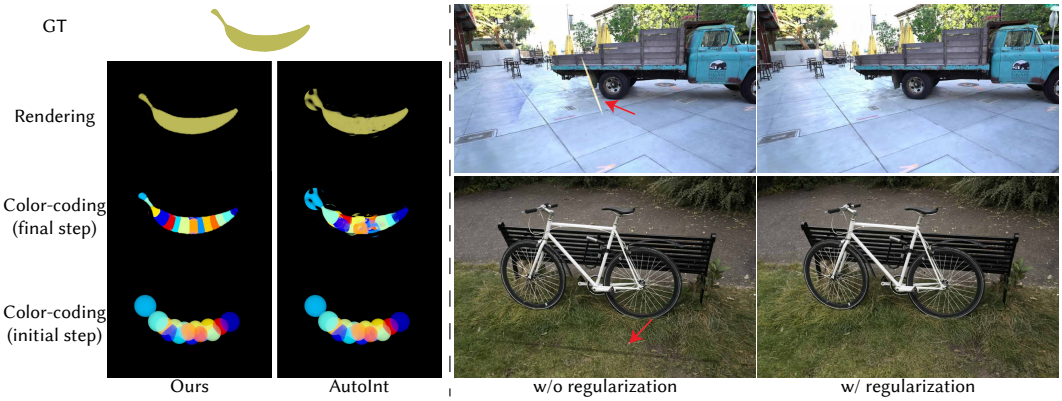

Figure 7: We analyze (left) the effect of an alternative neural integration strategy, AutoInt (Lindell et al., 2021), and (right) the effect of geometry regularization during training.

Results We summarize numerical results in Tab. 2. Our method achieves high-fidelity reconstructions with image quality and runtime comparable to state-of-the-art splatting-based approaches with analytic primitives, while generally requiring substantially less memory. Compared to monolithic neural representations, our neural splatting-based representation is more than an order of magnitude faster. While T-3DGS attains a similar trade-off, its control mechanisms are orthogonal to our contribution, which focuses on the representation itself; "taming" our neural primitives can be expected to yield significant gains as well. Fig. 6 confirms that our reconstructions are on par with the state of the art. In particular, our approach accurately captures fine-grained, structured geometry, such as the carpet region (highlighted in red) in the *Kitchen* and *Bonsai* scenes.

5 Ablation Studies

Here, we first investigate an alternative neural integration strategy compatible with a neural representation (Sec. 5.1). We then analyze the impact of the key parameters in our model formulation (Sec. 5.2) as well as the effect of the geometry regularization term (Sec. 5.3).

5.1 Neural Integration

AutoInt (Lindell et al., 2021) is an alternative approach for computing line integrals in a neural field, which we compare in Fig. 7, left. AutoInt uses a ray-based parameterization and applies automatic differentiation with respect to ray depth during training to obtain an integral network. However, this induces view-dependent density, leading to inconsistencies across viewpoints. In contrast, our method models the density field with a shallow network that depends only on 3D position, ensuring multi-view consistency.

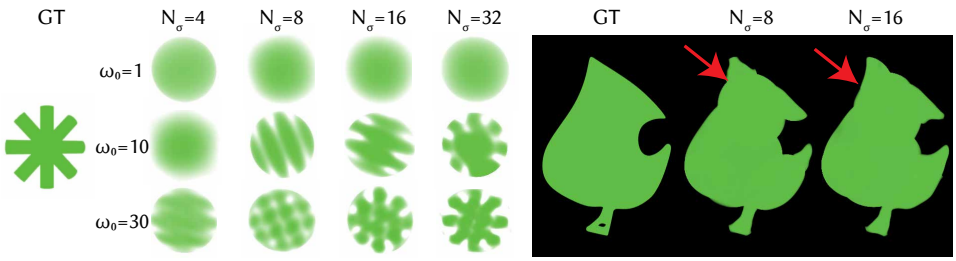

Figure 8: We visualize the effect of network width N_σ and frequency multiplier ω_0 on the expressivity of our neural density field.

5.2 Network Width

We analyze how the number of hidden neurons (N_σ) and the frequency multiplier (ω_0) affect the expressivity of our neural representation. We vary $N_\sigma \in 4, 8, 16, 32$ and $\omega_0 \in 1, 10, 30$, and run experiments on *Snowflake* using a single primitive and *Leaf* using eight primitives. As shown in Fig. 8, larger N_σ and higher ω_0 better reproduce the *Snowflake* structure and the smooth contours of *Leaf*. We further evaluate on real scenes from MipNeRF360, disabling densification and optimizing the same number of primitives with varying N_σ. With $N_\sigma = 4$, we obtain an average PSNR of 26.96; $N_\sigma = 8$ and $N_\sigma = 16$ yield 27.21 and 27.29, respectively. Although $N_\sigma = 16$ offers greater expressivity than 8 in toy settings, this advantage diminishes on real scenes due to the difficulty of optimizing a highly under-constrained problem. Balancing memory footprint and expressivity, we set $N_\sigma = 8$ and $\omega_0 = 30$ as the default configuration for all experiments.

5.3 Geometry Regularization

Jointly optimizing millions of neural primitives in complex scenes is highly under-constrained and prone to local minima, often resulting in extreme geometries, as shown in Fig. 7, right. We find that geometric regularization stabilizes training by penalizing elongated primitives. While numerical results remain similar, the regularization yields clear qualitative improvements.

6 Discussion and Conclusion

Our method is a novel radiance field representation that reconciles the expressivity of neural representations with the efficiency of splatting-based rendering techniques. We identify accurate density field integration as a key factor for achieving high expressivity in novel-view synthesis. Inspired by neural radiance fields, we formulate each primitive as a shallow network, which enables exact integration by evaluating the analytical anti-derivative with only two queries, reducing the ray-marching burden. While such a network has comparably limited representational capacity, the primitive-based approach mitigates the limitation by enabling a collection of tiny primitives to jointly reconstruct fine-grained scene details. This design provides both computational accuracy and efficiency. Furthermore, we show that ellipsoid-bounded neural primitives can be integrated into a differentiable splatting-based renderer, achieving real-time rendering performance. Our experiments demonstrate that neural primitives produce high-fidelity results comparable to 3DGS while requiring 10× fewer primitives and 6× fewer parameters, and delivering 100× speedups over neural-based methods. We believe that *splattable neural representation* opens new possibilities of integrating neural-based representation with splatting-based rendering techniques.

Although neural primitives exhibit substantial expressivity with limited memory resources, the complexity of the optimization landscape for millions of networks occasionally introduces convergence difficulties, hindering the expressivity of neural representations. A promising avenue for future research is to develop effective optimization or training strategies to fully unleash the expressivity of neural primitives. Moreover, as a general density field representation, our neural density field remains orthogonal to other techniques designed for color field and densification. Hence, integrating neural primitives with such advanced techniques is another interesting direction to explore in the future.

REFERENCES

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. *ICCV*, 2021.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023.

Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. Gs3: Efficient relighting with triple gaussian splatting. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–12, 2024.

Adam Celarek, George Kopanas, George Drettakis, Michael Wimmer, and Bernhard Kerbl. Does 3d gaussian splatting need accurate volumetric rendering? In *Computer Graphics Forum*, pp. e70032. Wiley Online Library, 2025.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16123–16133, 2022.

Haodong Chen, Runnan Chen, Qiang Qu, Zhaoqing Wang, Tongliang Liu, Xiaoming Chen, and Yuk Ying Chung. Beyond gaussians: Fast and high-fidelity 3d splatting with linear kernels, 2024. URL https://arxiv.org/abs/2411.12440.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. *arXiv:2502.01157*, 2025.

Chun Gu, Zeyu Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Tetrahedron splatting for 3d generation. In *NeurIPS*, 2024.

Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19812–19822, 2024.

Paul S Heckbert. Fundamentals of texture mapping and image warping. 1989.

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.

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 21360–21369, 2025b.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657428.

James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

Kaizhang Kang, Cihui Xie, Chengan He, Mingqi Yi, Minyi Gu, Zimin Chen, Kun Zhou, and Hongzhi Wu. Learning efficient illumination multiplexing for joint capture of reflectance and shape. *ACM Trans. Graph.*, 38(6):165–1, 2019.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

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.

Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. *arXiv preprint arXiv:2406.02720*, 2024.

Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21136–21145, 2024.

David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14556–14565, 2021.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–11, 2025.

Steffan Lloyd, Rishad A Irani, and Mojtaba Ahmadi. Using neural networks for fast numerical integration and optimization. *IEEE Access*, 8:84519–84531, 2020.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021.

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*, pp. 20654–20664, 2024.

Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis. *arXiv preprint arXiv:2410.01804*, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

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*, pp. 7210–7219, 2021.

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

Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. SplatFields: Neural gaussian splats for sparse 3d and 4d reconstruction. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

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.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *International Conference on Computer Vision (ICCV)*, 2021.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325, 2021.

Kartic Subr. Q-net: A network for low-dimensional integrals of neural proxies. In *Computer Graphics Forum*, volume 40, pp. 61–71. Wiley Online Library, 2021.

Chinmay Talegaonkar, Yash Belhe, Ravi Ramamoorthi, and Nicholas Antipa. Volumetrically consistent 3d gaussian rasterization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10953–10963, 2025.

Gregory H Teichert, Anirudh R Natarajan, Anton Van der Ven, and Krishna Garikipati. Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions. *Computer Methods in Applied Mechanics and Engineering*, 353:201–216, 2019.

Nicolas von Lützow and Matthias Nießner. Linprim: Linear primitives for differentiable volumetric rendering. *arXiv preprint arXiv:2501.16312*, 2025.

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.

Lina Yan, Jingjing Di, and Ke Wang. Spline basis neural network algorithm for numerical integration. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 7(3):458–461, 2013.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5752–5761, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zeng Zhe-Zhao, Wang Yao-Nan, and Wen Hui. Numerical integration based on a neural network algorithm. *Computing in science & engineering*, 8(4):42–48, 2006.

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pp. 29–538. IEEE, 2001.

Matthias Zwicker, Jussi Rasanen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. Perspective accurate splatting. In *Proceedings-Graphics Interface*, pp. 247–254, 2004.