RayGauss: Volumetric Gaussian-Based Ray Casting for Photorealistic Novel View Synthesis

Hugo Blanc, Jean-Emmanuel Deschaud, Alexis Paljic
MINES Paris - PSL
France
{hugo.blanc,jean-emmanuel.deschaud,alexis.paljic}@minesparis.psl.eu

Abstract

Differentiable volumetric rendering-based methods made significant progress in novel view synthesis. On one hand, innovative methods have replaced the Neural Radiance Fields (NeRF) network with locally parameterized structures, enabling high-quality renderings in a reasonable time. On the other hand, approaches have used differentiable splatting instead of NeRF's ray casting to optimize radiance fields rapidly using Gaussian kernels, allowing for fine adaptation to the scene. However, differentiable ray casting of irregularly spaced kernels has been scarcely explored, while splatting, despite enabling fast rendering times, is susceptible to clearly visible artifacts. Our work closes this gap by providing a physically consistent formulation of the emitted radiance c and density σ, decomposed with Gaussian functions associated with Spherical Gaussians/Harmonics for all-frequency colorimetric representation. We also introduce a method enabling differentiable ray casting of irregularly distributed Gaussians using an algorithm that integrates radiance fields slab by slab and leverages a BVH structure. This allows our approach to finely adapt to the scene while avoiding splatting artifacts. As a result, we achieve superior rendering quality compared to the state-of-the-art while maintaining reasonable training times and achieving inference speeds of 25 FPS on the Blender dataset. Associated code will be released publicly on GitHub.

1. Introduction

Novel View Synthesis (NVS), which combines existing views of a scene to generate images from unknown viewpoints, saw significant advancements with the publication of Neural Radiance Fields (NeRF) [16]. This novel approach leverages a differentiable physical rendering algorithm [14], to learn radiance fields through a neural network, enabling the generation of high-quality photorealistic images. Since then, several works aimed at improving NeRF's performance, some focusing on finding more efficient representations of radiance fields in terms of computation time and rendering quality. Various data structures were investigated to substitute the neural network and represent the radiance fields: voxel grids [5] [25] [17], sets of tetrahedra [12], point clouds [27] [8]. These structures have the advantage of enabling the representation of radiance fields using local parameters, thereby reducing computation times by limiting field evaluation to local parameters instead of relying on a neural network representing the entire scene. These methods also aim to enhance rendering quality by allowing a more precise representation of radiance fields through data structures better suited to the scene's geometry.

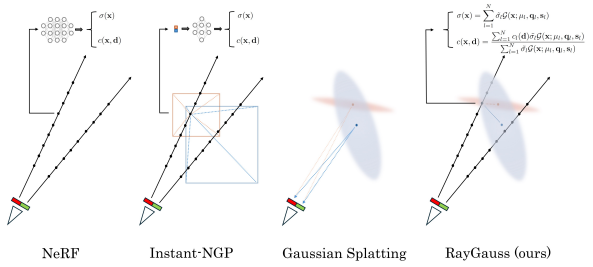

Figure 1. Our method (RayGauss) compared to the main radiance field rendering methods

Our work follows this approach by proposing a novel definition of radiance fields based on their decomposition into a weighted sum of locally supported elliptical basis functions with optimizable positions, orientations, and scales. Unlike voxel grid-based methods, this allows for finer adaptation to the scene's geometry without limiting representation resolution. Furthermore, a combination of spherical harmonics (SH) and spherical Gaussians (SG) is associated with these basis functions to better represent high-frequency color variations such as specular reflections. Additionally, we leverage the volume ray casting approach, which generates high-quality images with less restrictive

simplifications compared to splatting-based approaches like 3D Gaussian Splatting [8]. Indeed, splatting algorithms simplify the scene using multiple approximations, allowing faster rendering but causing many artifacts when deviating from the training views, such as flickering due to a sudden change in primitives ordering [21]. For these reasons, our approach achieves higher rendering quality by combining the best of both worlds: Firstly, radiance fields are represented by elliptical functions associated with efficient SH/SG radiance parameters that adapt very well to the scene geometry and appearance. Secondly, the Volume Raycasting algorithm avoids the artifacts of Splatting, such as flickering.

Furthermore, implementing the Raycasting algorithm with sparse primitives is challenging because evaluating the radiance fields at a given point requires knowing which primitives/basis functions contribute to the calculations, which is much less straightforward than when using a voxel grid, for example. Therefore, we introduce a specialized ray casting algorithm for sparse basis functions that sequentially integrates the radiance field slab-by-slab, meaning it accumulates colorimetric properties along the ray in slab of space corresponding to multiple samples, rather than processing individual samples [9]. The implementation of this algorithm and its backpropagation relies on a Bounding Volume Hierarchy (BVH) implemented on GPU with the OptiX library. This allows interactive rendering times and reasonable training times. Thus, our contributions are as follows:

- Volume Ray Casting of radiance fields decomposed into elliptical basis functions combined with SH/SG for efficient adaptation to scene geometry and appearance.
- Efficient implementation of Volume Ray Casting and its backpropagation using a slab-by-slab integration algorithm along the ray and leveraging a Bounding Volume Hierarchy for fast ray-ellipsoïd intersection.

2. Background and Related Work

We will introduce the theoretical basis of radiative transfer due to its essential role in our method and modern NVS approaches. Next, we will discuss the NeRF method and subsequent research on scene representation. Finally, we will discuss 3D Gaussian Splatting [8] and its flexible scene representation using Gaussians while highlighting the weaknesses of the splatting algorithm.

Radiative Transfer and Volume Rendering: Recent approaches to Novel View Synthesis, such as NeRF [16], Instant-NGP [17], and 3D Gaussian Splatting [8], leverage a differentiable volume rendering equation to optimize scene representation from images (Fig. 1). Their rendering equations are derived from the radiative transfer equation in an absorbing and emitting medium, which models the variation in radiance as it travels through an infinitesimal volume element at position $\mathbf{x}$ in direction ω [14] [30]:

$$(\omega \cdot \nabla)L(\mathbf{x},\omega) = -\sigma(\mathbf{x})L(\mathbf{x},\omega) + \sigma(\mathbf{x})c(\mathbf{x},\omega) \quad (1)$$

where $L(\mathbf{x},\omega)$ is the radiance at position $\mathbf{x}$ in direction ω, $\sigma(\mathbf{x})$ the absorption coefficient and $c(\mathbf{x},\omega)$ the emitted radiance [3]. The first term quantifies the absorption of radiance, and the second term its emission. The solution to this non-homogeneous linear differential equation is:

$$L(\mathbf{x},\omega) = \int_0^\infty c(\mathbf{y},\omega)\sigma(\mathbf{y})T(\mathbf{x},\mathbf{y})\,d\mathbf{y} \quad (2)$$

by integrating along the ray reaching $\mathbf{x}$ in the direction ω defined by the points $\mathbf{y}(t) = \mathbf{x} - t\omega$, and where $T(\mathbf{x},\mathbf{y}) = e^{-\int_0^t \sigma(\mathbf{x}-s\omega)\,ds}$ is the transmittance between $\mathbf{x}$ and $\mathbf{y}$. This expression appears in the literature of Volumetric Light Transport Simulation [18]. In the context of volumetric rendering, a modified parameterization is used: fictive rays denoted by $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ originate from the camera center $\mathbf{o}$ and traverse the scene opposite to the light direction, assimilating $\mathbf{d}$ to $-\omega$. In computer vision, the focus is on the ray's color, interpreting radiance L as the color C of the ray, and the absorption coefficient σ renamed density. This parameterization leads to the classic volumetric rendering equation [16] [30]:

$$\begin{aligned} C(\mathbf{r}) &= \int_0^\infty c(\mathbf{r}(t),\mathbf{d})\sigma(\mathbf{r}(t))T(t)\,dt, \\ T(t) &= e^{-\int_0^t \sigma(\mathbf{r}(s))\,ds} \end{aligned} \quad (3)$$

This equation is the basis for recent Novel View Synthesis approaches like NeRF and 3D Gaussian Splatting.

NeRF and Scene Representation: NeRF optimizes a neural network representing the fields c and σ using training images and their corresponding camera poses [16]. During training, images are rendered by ray-casting from training poses and compared to ground truth using a loss function, allowing the update of network parameters via gradient descent. During inference, the trained network generates realistic images from new camera poses by ray casting.

More precisely, NeRF uses the differentiable Volume Ray Casting algorithm inspired by Max computations in [14]. It consists in launching rays in the 3D space to compute a discretized version of equation 3 using N samples $\tilde{t_i}$ along the ray:

$$\begin{aligned} C(\mathbf{r}) &= \sum_{i=0}^{N} \left(1 - \exp(-\sigma_i \Delta t)\right) c_i T_i \\ T_i &= \exp\left(-\sum_{j=0}^{i-1} \sigma_j \Delta t\right) \end{aligned} \quad (4)$$

where Δt the discretization step, and the i-th field values denoted as $\sigma_i = \sigma(\mathbf{r}(\tilde{t_i}))$ and $c_i = c(\mathbf{r}(\tilde{t_i}), \mathbf{d})$, this amounts to assuming that the fields are piecewise constant in samples neighborhood. The reader interested in the computational details can find more information in [14] [15].

Thus, NeRF's contribution is using a differentiable volumetric rendering algorithm to learn the scene parameters represented by a neural network. Subsequent research papers have proposed improvements to the scene representation by using different structures to store scene parameters locally, thus eliminating the dependency on a computationally costly neural network representing the entire scene and improving the appearance by better adapting the local parameters than a global MLP. Moreover, due to their sparse nature, some data structures can apply classic ray-casting acceleration strategies such as empty space skipping [6]. Among these structures, one can notably mention voxel grids [25], grids with hash encoding [17], grids of small MLPs [22], sets of tetrahedrons [12], and point clouds [27]. Our hypothesis is that point clouds are best suited to adapt to scenes. Each point can host a locally supported function, such as Gaussian functions [8], representing local scene details. Optimizing these functions' positions and shapes enables precise adaptation to scene geometry, unlike voxel grids constrained by their resolution. Furthermore, to the best of our knowledge, no method currently allows for optimizing a scene represented only by irregularly spaced basis functions with the ray-casting algorithm without coarse approximations. The approach in Point-NeRF [27] is interesting but requires using a MVSNet, which proves cumbersome. Additionally, its expression of c and σ involves using a neural network and only considers the eight closest neighbors in its calculations. Moreover, as discussed in the following section, the 3D Gaussian Splatting method and numerous point cloud rendering methods [29] make the splatting approximation, which can cause visible artifacts such as flickering when changing viewpoints. This choice is due to the scarcity of algorithms that efficiently perform ray casting on local basis functions associated with points.

Gaussian Splatting: 3D Gaussian Splatting [8] made a breakthrough by using a differentiable splatting algorithm to render a scene represented by Gaussians. It draws inspiration from the classic Elliptical Weighted Average (EWA) Splatting approach [30]. The main advantages of this approach are the use of Gaussians, which allow for efficient adaptation to the scene geometry, and an efficient tile-based rasterizer, which enables fast rendering of the scene. Starting from a set of 3D Gaussians associated with color parameters c_i and opacity α_i, the splatting algorithm projects the 3D Gaussians into the camera space, performs a global sorting by their depth, and sequentially adds their contribution from the nearest to the farthest for each pixel color C using the equation:

$$
\begin{aligned}
C &= \sum_{i \in \mathcal{N}} c_i \alpha'_i \prod_{j=1}^{i-1} (1 - \alpha'_j) \\
\alpha'_i &= \alpha_i \times \exp\left(-\frac{1}{2} (x' - \mu'_i)^\top \Sigma'^{-1}_i (x' - \mu'_i) \right)
\end{aligned}
\tag{5}
$$

Where x' the pixel coordinates in the camera frame, μ'_i and Σ'_i are the mean and covariance of the Gaussian projected into the camera frame, and $\mathcal{N}$ denotes the set of Gaussians. Interested readers can find the implementation details in the original paper [8] and the following review [4]. Like the discretized equation of ray casting, the formulation of splatting 5 is also an approximation of Eq. 3. However, the approximations used differ: Gaussians are assumed to be non-overlapping, which does not hold for complex surfaces. Indeed, multiple overlapping Gaussians are required for precise scene geometry reconstruction. Then, each Gaussian contributes at most once per ray, aggregated at a distance corresponding to the depth of the Gaussian (Gaussian mean projected along the viewing direction). Also, the sequential addition of Gaussian contributions based on global depth is an approximation. It does not account for the proper intersection between the Gaussians and a given ray, further reducing the coherence of the representation. Finally, the projection of Gaussians into the camera frame is also an approximation introduced in [30]. Thus, in our approach, we use the ray casting algorithm to suffer fewer approximations because we hypothesize that, given a high enough sampling rate, it maintains better coherence in scene representation during training and inference and should yield better graphical results.

3. Scene Representation

Our rendering model is based on Eq. 4. Thus, the performance of our approach greatly depends on the formulation of our functions σ and c. The main idea of our work is to approximate these fields through a decomposition using irregularly spatially distributed basis functions while maintaining physical coherence. This will allow for better adaptation to the geometry of the represented scene without resolution limits. Thus, for a set of basis functions $\{\phi_l, \psi_l\}_{l=1,..,N}$, our fields will be of the form:

$$
\begin{aligned}
\sigma(\mathbf{x}) &= \sum_{l=1}^{N} w_{\sigma_l} \phi_l(\mathbf{x}) \\
c(\mathbf{x}, \mathbf{d}) &= \sum_{l=1}^{N} w_{c_l}(\mathbf{d}) \psi_l(\mathbf{x})
\end{aligned}
\tag{6}
$$

In what follows, we will explain the choices of weights and basis functions we made to best approximate the scene with the fields σ and c while maintaining physical consistency in the σ/c relationship.

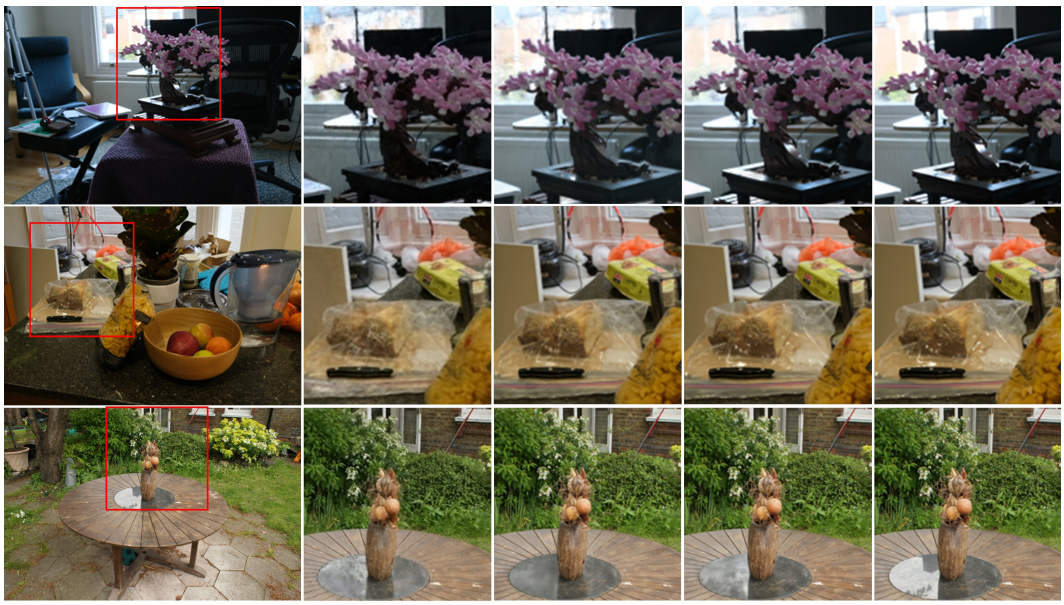

Figure 2. **Test set images from the Mip-NeRF 360 Dataset**

3.1. Irregularly Spatially-Distributed Emissive and Absorptive Primitives

As we will see, the expression previously given in Equation 6 can be physically assimilated to the case of a scene composed of N independent emissive and absorbing entities, each distributed over the support of their basis function with their dedicated density σ_l and emissive radiance c_l [24] [11]. Indeed, given a basis function ϕ_l for $l \in [1, N]$, the evolution of the density σ_l of the l-th entity in the scene is such that:

$$\sigma_l(\mathbf{x}) = \tilde{\sigma_l} \cdot \phi_l(\mathbf{x}) \tag{7}$$

where $\tilde{\sigma_l}$ is the maximum density with $0 < \phi_l(\mathbf{x}) < 1$. On the other hand, we choose an emitted radiance c that depends only on the direction of emission $\mathbf{d}$ and is constant in space.

$$c_l(\mathbf{x}, \mathbf{d}) = c_l(\mathbf{d}) \tag{8}$$

This representation allows us to depict objects whose appearance changes according to the viewing direction. We will develop this aspect further.

Then, to derive a physically coherent total density σ and total emitted radiance term c for the scene, we can revisit the radiative transfer equation 1 for the case of N independent entities with distinct density $\sigma_l(\mathbf{x})$ and emissive radiance c_l, in which case it is expressed as follows:

$$(\mathbf{d} \cdot \nabla) L(\mathbf{x}, \mathbf{d}) = -\left(\sum_{l=1}^{N} \sigma_l(\mathbf{x}) \right) L(\mathbf{x}, \mathbf{d}) + \sum_{l=1}^{N} \sigma_l(\mathbf{x}) c_l \tag{9}$$

where: $(\mathbf{d} \cdot \nabla) L(\mathbf{x}, \mathbf{d})$ is the rate of change of radiance $L(x, \mathbf{d})$ [24] [11]. The previous radiative transfer equation allows for considering variations in radiance due to absorption phenomena (first term) and emission phenomena (second term) due to each independent entity. We recall that the radiative transfer equation for a medium described by global density function $\sigma(\mathbf{x})$ and global emitted radiance field $c(\mathbf{x}, \mathbf{d})$ is given by Eq. 1. Thus, we deduce that it is possible to place ourselves within this framework by defining the density field as follows:

$$\sigma(\mathbf{x}) = \sum_{l=1}^{N} \sigma_l(\mathbf{x}) = \sum_{l=1}^{N} \tilde{\sigma_l} \phi_l(\mathbf{x}) \tag{10}$$

Furthermore, the emitted radiance field can then be defined as:

$$c(\mathbf{x}, \mathbf{d}) = \frac{\sum_{l=1}^{N} c_l(\mathbf{d}) \tilde{\sigma_l} \phi_l(\mathbf{x})}{\sum_{l=1}^{N} \tilde{\sigma_l} \phi_l(\mathbf{x})} \tag{11}$$

As initially intended, we find a formulation with a weighted sum of basis functions. Moreover, this formulation allows us to maintain consistent physical behavior during rendering, avoiding some visual artifacts while using a flexible basis function.

3.2. Selection of the basis function

The base function class must allow our primitives to adapt optimally to the scene's geometry. In particular, we study radial and elliptical basis functions to leverage their approximation power [19]. Thus, each basis function ϕ_l is

centered at μ_l and evolves with $r = d(\mathbf{x}, \mu_l)$, where d is the Euclidean distance for radial functions or the Mahalanobis distance for elliptical functions. Moreover, we restrict ourselves to decreasing functions with respect to r, allowing us to use point clouds representing a raw depiction of the scene as an initialization for the center μ_l of the basis functions. Indeed, the positions of the points provide a nice prior for regions that highly interact with light.

More specifically, we have chosen anisotropic Gaussian functions as our basis function class, as this gives the best results among the functions studied (see Tab. 4). A given basis function can thus be expressed as follows:

$$\mathcal{G}(\mathbf{x}; \mu, \mathbf{\Sigma}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \quad (12)$$

Where μ is the mean vector, $\mathbf{\Sigma}$ is the covariance matrix. Here, we can notice that $\sqrt{(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)}$ is the Mahalanobis distance associated with the matrix Σ. To achieve a more suitable parameterization, the covariance matrix $\mathbf{\Sigma}$, being positive and symmetric, can be decomposed according to the spectral theorem into a rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$ such that:

$$\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$$

Additionally, the rotation matrix can be expressed in terms of quaternions $\mathbf{q} \in \mathbb{R}^4$ such that $\mathbf{R} = \mathbf{R}(\mathbf{q})$, and the scaling matrix $\mathbf{S}$ can be expressed in terms of $\mathbf{s} = (s_x, s_y, s_z)$ such that $\mathbf{S} = \text{diag}(s_x, s_y, s_z)$. Thus, we will refer to our Gaussian basis functions as $\mathcal{G}(\mathbf{x}; \mu, \mathbf{q}, \mathbf{s})$.
Therefore, we can now describe our density field $\sigma(\mathbf{x})$ and emitted radiance field $c(\mathbf{x}, \mathbf{d})$ as follows:

$$\begin{aligned} \sigma(\mathbf{x}) &= \sum_{l=1}^{N} \tilde{\sigma}_l \mathcal{G}(\mathbf{x}; \mu_l, \mathbf{q}_l, \mathbf{s}_l) \\ c(\mathbf{x}, \mathbf{d}) &= \frac{\sum_{l=1}^{N} c_l(\mathbf{d})\tilde{\sigma}_l \mathcal{G}(\mathbf{x}; \mu_l, \mathbf{q}_l, \mathbf{s}_l)}{\sum_{l=1}^{N} \tilde{\sigma}_l \mathcal{G}(\mathbf{x}; \mu_l, \mathbf{q}_l, \mathbf{s}_l)} \end{aligned} \quad (13)$$

3.3. Direction-dependent emitted radiance

To correctly represent objects whose color appearance varies according to the direction of observation, we made the emitted radiance of our primitives dependent on the viewing direction. One parameterization that gave the best results was a combination of spherical harmonics and spherical Gaussians, such that emitted radiance of l-th primitive is:

$$c_l(\mathbf{d}) = c_{\text{low},l}(\mathbf{d}) + c_{\text{high},l}(\mathbf{d}) \quad (14)$$

where:

$$\begin{aligned} c_{\text{low},l}(\mathbf{d}) &= \sum_{j=0}^{L_1} \sum_{m=-j}^{j} \tilde{c}_{l,jm} Y_{jm}(\mathbf{d}) \\ c_{\text{high},l}(\mathbf{d}) &= \sum_{j=0}^{L_2} k_{l,j} e^{\lambda_{l,j}(\mathbf{d}\cdot\mathbf{p}_{l,j}-1)} \end{aligned} \quad (15)$$

$Y_{jm}(\mathbf{d})$ is the spherical harmonic of degree j and order m associated with its coefficient $\tilde{c}_{l,jm} \in \mathbb{R}^3$, while $k_{l,j} \in \mathbb{R}^3$ is the coefficient of the j-th spherical gaussian with lobe sharpness $\lambda_{l,j} \in \mathbb{R}$ and lobe direction $\mathbf{p}_{l,j} \in \mathbb{R}^3$.

This choice is motivated by the fact that low-degree Spherical Harmonics are better suited to represent low-frequency variations. In contrast, Spherical Gaussians can represent any frequency, including high frequencies. Thus, their combined use allows for representing both low and high-frequency phenomena. Similar considerations can be found for different applications in articles such as MixLight [7].

4. Scene initialization and optimization

Our goal is to use the scene representation discussed earlier and optimize its parameters $P = \{(\tilde{\sigma}_l, \tilde{c}_l, \mu_l, \mathbf{q}_l, \mathbf{s}_l) \mid l = 1, \ldots, N\}$, where $\tilde{c}_l$ denotes the set of colorimetric parameters of the l-th basis function, $\tilde{\sigma}_l$ its density parameter, $\mu_l, \mathbf{q}_l, \mathbf{s}_l$ the parameters determining the shape of the base function. Once the parameters are optimized, the objective is to infer images that faithfully represent the scene.

4.1. Scene initialization

To initialize this representation, we leverage sparse point clouds obtained via Structure from Motion (SfM), where the position and color of these points are used to initialize colorimetric parameters and the center of the basis functions μ_l. Additionally, the quaternions are initialized to match the identity matrix, and the scale parameters are initialized to the average distance from each point to its three nearest neighbors.

4.2. Scene Optimization

Once the representation is initialized, we optimize it using a stochastic gradient descent algorithm. To do this, we rely on a set of training images, where each image is associated with the pose of its camera. Iteratively, over the training set, we generate the scene image from the current camera pose and compare it to the current training image using a given loss function. Subsequently, we compute gradients of the loss function with respect to the scene parameters, allowing us to update them by backpropagation. In particular, the loss function L is a weighted sum of the L_1 loss function and the structural dissimilarity (DSSIM):

$$L = (1-\lambda)\cdot L_1 + \lambda \cdot \text{DSSIM}$$

where λ is a weighting factor between the L1 loss and DSSIM, we follow best practices developed in 3D Gaussian Splatting [8]. Furthermore, we are in a similar case to 3D Gaussian Splatting regarding adaptive control of primitives. Indeed, we also deal with primitives irregularly distributed

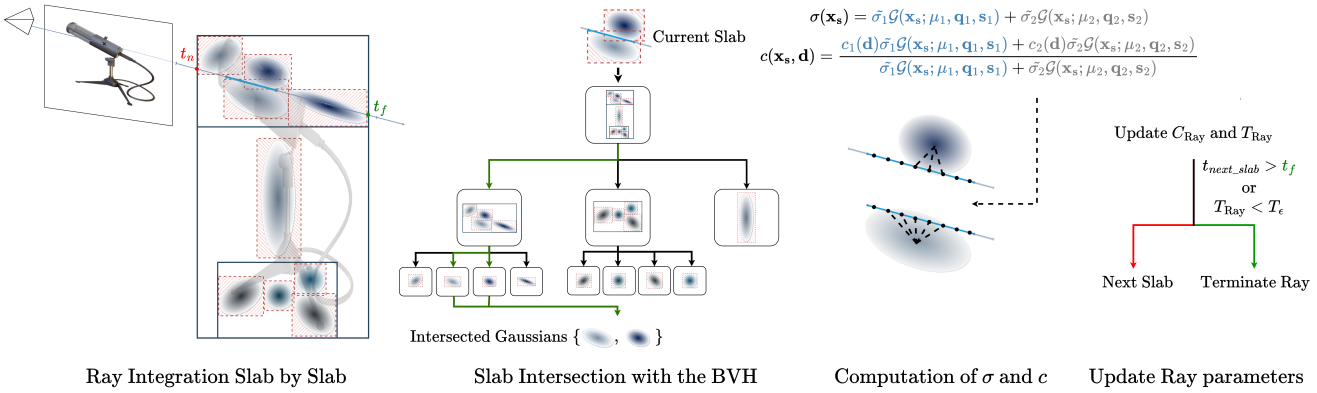

Figure 3. **Overview of our slab-by-slab Ray Casting algorithm:** When a given ray intersects the scene's bounding box, the ray is processed by successive slabs. For a given slab, we check the intersection with the Gaussians using the BVH. Then, the attributes of the slab are calculated to update the current color and transmittance of the ray.

in space. For this reason, we rely on the point cloud densification heuristics described in [8]. Regarding removing outlier points, we eliminate points whose density falls below a fixed threshold σ_ϵ. Additionally, we progressively unlock the ability to learn different colorimetric parameters to enforce model coherence at low frequencies before moving to higher frequencies. Thus, we gradually unlock the degrees of spherical harmonics and then unlock spherical Gaussians.

5. Spatially irregularly distributed base functions ray casting

This approach has been scarcely explored due to the major obstacle of implementing an algorithm that enables efficient differentiable volume ray casting for a scene defined with irregularly distributed basis functions. Firstly, it should be noted that since the support of our basis functions is unbounded in the chosen case (Gaussian functions), it would potentially require evaluating all basis functions to compute σ and c at any spatial position. This promises to slow down the rendering of our images significantly. To mitigate this, we truncate our basis functions to a local domain. The criterion retained to determine this domain for each primitive is as follows:

$$\sigma_l(\mathbf{x})_{\text{approx}} = \begin{cases} \sigma_l(\mathbf{x}) & \text{if } \sigma_l(\mathbf{x}) \geq \sigma_\epsilon \\ 0 & \text{if } \sigma_l(\mathbf{x}) < \sigma_\epsilon \end{cases} \tag{16}$$

where σ_ϵ is a fixed threshold, this amounts to disregarding areas with a density coefficient below a given threshold. These areas have minimal interaction with light and, therefore, have little influence on the resulting color. Thus, for a low threshold, this approximation appears acceptable. Now that the influence of each primitive is confined to a specific domain, we implement a differentiable algorithm capable of computing an image through volume ray casting with radial or elliptical basis functions truncated to a finite domain. We rely on a bounding volume hierarchy (BVH) to achieve this, drawing inspiration from the work conducted by [10]. Thus, to compute the color associated with a given ray, we construct a BVH containing ellipsoids if the chosen base is elliptical or spheres if it is radial. These geometric figures represent the level set associated with the previous truncation. Once the acceleration structure has been built, we launch each ray in parallel in the acceleration structure, in practice using the Nvidia OptiX framework. Integration along the ray is done sequentially by slabs of samples, each containing a fixed number of samples (see Fig. 3). By doing this, our approach avoids the drawbacks of naive methods: integrating sample-by-sample is slow due to repeated neighborhood calculations. While, calculating σ and c attributes for all samples in one pass limits the number of samples, requires a large buffer, and prevents early ray termination. Our method sequentially calculates neighborhoods for entire slabs and stores slab attributes in a small fixed-size buffer, reducing costly calculations and allowing efficient early ray termination at the slab level.

6. Implementation and Experiments

6.1. Implementation

Our implementation relies on the PyTorch library for managing the optimization framework, combined with a Python binding of Nvidia Optix [20] using the Cupy library. Thus, the computationally intensive parts of our algorithm, including BVH management, ray casting, and derivative calculations, are implemented with the Optix library and custom CUDA kernels. Details on our optimization parameters can be found in supplementary materials.

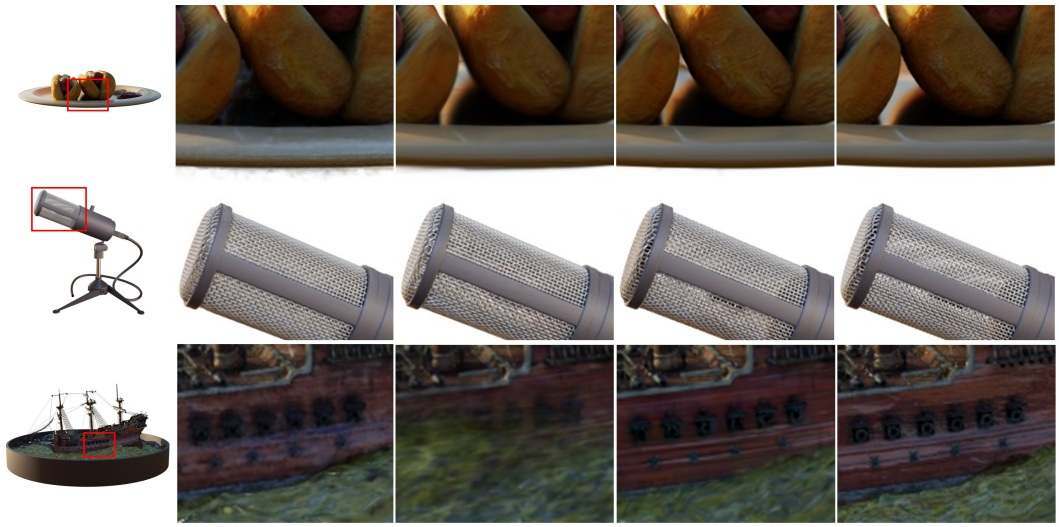

Figure 4. **Test set images from the Blender Dataset**

6.2. Results and Evaluation

To quantify our method's performance, we tested it on two widely used datasets in the NVS domain: the Blender dataset [16] and the Mip-NeRF 360°dataset [1]. This allows us to test our method across a wide range of situations. The Blender dataset contains 8 scenes with complex objects exhibiting non-Lambertian effects. Each scene consists of 100 training images, 200 test images, and the corresponding exact camera parameters for 360° views of the object. The Mip NeRF 360° dataset comprises 9 scenes, 5 outdoor and 4 indoor, each featuring a central object surrounded by a complex background. Each scene is accompanied by images with camera parameters estimated by COLMAP [23]. We evaluate our method by downscaling images by a factor of 8 for this dataset due to large image size. Moreover, we compare our results primarily with 3D Gaussian Splatting [8] and subsequent research papers such as Mip-Splatting [28] due to the similarity in the representation used.

Results on the Blender dataset: For synthetic scenes in the Blender dataset, we provide quantitative comparisons in Tab. 1 and qualitative comparisons in Fig. 4 with state-of-the-art methods. For the results in Tab. 1, the values used are those reported in the state-of-the-art methods, except for those denoted with an *, which have been recalculated using the available code.

Quantitative results show that our approach surpasses state-of-the-art results on most individual scenes. Additionally, we achieve an average PSNR of **34.34 dB**, attesting our model's ability to finely adapt to the geometry and appearance of the scenes compared to other state-of-the-art methods. Qualitative results also support this, for example, with a boat hull better reconstructed in the *ship* scene compared to 3D Gaussian Splatting reconstruction (Figure 4).

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	**Avg.**
NeRF [16]	34.17	25.08	30.39	36.82	33.31	30.03	34.78	29.30	31.74
Zip-NeRF [2]	34.84	25.84	33.90	37.14	34.84	31.66	35.15	31.38	33.10
Instant-NGP [17]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
Mip-NeRF360 [1]	35.65	25.60	33.19	37.71	36.10	29.90	36.52	31.26	33.24
Point-NeRF [27]	35.40	26.06	36.13	37.30	35.04	29.61	35.95	30.97	33.30
Gaussian Splatting [8]*	35.85	26.22	35.00	37.81	35.87	30.00	35.40	30.95	33.39
Mip-Splatting [28]*	36.03	26.29	35.33	37.98	36.03	30.29	35.63	30.50	33.51
PointNeRF++ [26]	36.32	26.11	34.43	37.45	36.75	30.32	36.85	31.34	33.70
RayGauss (ours)	36.89	27.19	35.00	38.07	36.62	31.35	37.58	32.01	34.34

Table 1. **PSNR scores for Blender dataset.**

Results on the Mip-NeRF 360 Dataset: Regarding Mip-NeRF 360, we provide quantitative results in Tab. 2 and qualitative comparisons in Fig. 2. A general trend we observe is that our approach performs better on most indoor scenes than current state-of-the-art methods such as Mip-Splatting [1] and slightly worse on outdoor scenes. This may be due to the adaptive control of gaussians strategy, which adapts well in the case of Gaussian Splatting [8] but may perform worse with our approach. Nonetheless, we achieved better overall results across all scenes with an average PSNR of 29.96 dB, which is **+0.81 dB** higher than Gaussian Splatting.

	Indoor				**Outdoor**					
	bonsai	counter	kitchen	room	bicycle	flowers	garden	stump	treehill	**Avg.**
NeRF [16]*	22.10	22.34	22.00	24.46	19.35	19.49	22.70	21.43	21.02	21.65
Instant-NGP [17]*	27.04	24.25	23.44	27.30	23.69	21.41	25.64	22.56	22.22	24.17
Gaussian Splatting [8]*	33.42	30.21	33.40	32.95	27.33	23.71	29.58	27.78	24.00	29.15
Mip-Splatting [28]*	33.44	30.43	34.30	33.30	27.62	23.79	29.78	27.89	24.25	29.42
RayGauss (ours)	35.22	31.83	34.46	33.28	27.35	23.63	29.87	26.94	24.04	29.62

Table 2. **PSNR scores for Mip-NeRF 360 dataset.** All methods are trained and tested on downsampled images by a factor of 8.

Performance Analysis: We discuss here results in Tab.

3, which show the performance of our approach compared to Gaussian Splatting. These metrics were obtained using an RTX 4090 GPU with 24GB of RAM on 800x800 pixel images. Our approach has slower training and inference times compared to Gaussian Splatting due to the computational cost of ray casting. Generally, our computation times depend predominantly on the number of rays, while Gaussian Splatting's times are influenced by the number of projected primitives. However, our training times remain reasonable (30 minutes per scene on the Blender dataset) and real-time inference is achieved on Blender, although slower on Mip-NeRF 360. We also note that our method generates fewer primitives for better visual quality. Thus, Gaussian Splatting likely adds more Gaussians to compensate for splatting artifacts.

	Number of Gaussians	Training Time	Inference Time
Blender dataset			
Gaussian Splatting [8]*	295k	10 min	400 FPS
RayGauss (ours)	205k	30 min	25 FPS
Mip-NeRF 360			
Gaussian Splatting [8]*	2.7M	15 min	130 FPS
RayGauss (ours)	2.3M	2h 30 min	5 FPS

Table 3. **Comparison of Gaussian Splatting and RayGauss performance on Blender and Mip-NeRF 360 datasets.** Numbers are averaged per scene on each dataset.

6.3. Ablation Studies

Method \Metrics	PSNR ↑	SSIM ↑
(Gaussian Splatting [8]) Spherical Harmonics*	33.39	0.969
(Ours) Spherical Harmonics	33.69	0.972
(Ours) Isotropic Gaussian	33.44	0.969
(Ours) Anisotropic Gaussian	34.34	0.974
(Ours) Anisotropic Inverse Multiquadric	30.84	0.962
(Ours) Anisotropic Inverse Quadratic	32.95	0.967
(Ours) Anisotropic C^0 Matérn	33.83	0.972
(Ours) Anisotropic Bump	34.18	0.972
(Ours) Anisotropic Wendland	34.09	0.972
(Ours) Anisotropic Gaussian	34.34	0.974
(Ours) RGB	30.38	0.961
(Ours) Spherical Harmonics	33.69	0.972
(Ours) Spherical Gaussians	34.11	0.972
(Ours) Spherical Gaussians + Spherical Harmonics	34.34	0.974

Table 4. **Evaluation of different aspects of our method:** Rendering Algorithm, Basis Function, and Colorimetric Parameters by averaging PSNR and SSIM scores on Blender Dataset

In this section, we quantify the influence of the different components of our approach. Thus, we test different configurations by averaging our PSNR and SSIM results on the NeRF-Synthetic dataset. Results are summarized in Tab. 4. Additionally, we prioritize PSNR analysis to assess the different approaches, as this metric has been found more effective than SSIM or LPIPS according to recent studies [13].

3D Gaussian Splatting: We first compare our method with Gaussian basis function and colorimetric parameters similar to 3D Gaussian Splatting (Spherical Harmonics) and observe that we achieve better results with an advantage of **+0.3 dB**, demonstrating that our ray-casting algorithm enables superior rendering quality than splatting while also avoiding rendering artifacts such as flickering. This can be observed in the supplementary videos, which show the flickering in Gaussian Splatting that is absent in our approach.

Basis function: We then conduct tests to confirm the choice of the Gaussian function. It can be observed that the anisotropic Gaussian yields better results than its isotropic counterpart, with an average difference of **+0.9 dB**. Various classical basis functions were also tested, including those with locally native support (such as the Bump function) or globally supported (such as the Inverse Quadratic function). Several showed promising results, notably the Wendland function. However, the anisotropic Gaussian performs best with at least a **+0.25 dB** difference (see Tab. 4). This justifies the choice of the anisotropic Gaussian as the basis function.

Colorimetric parameters: Then, we study the influence of the parameterization of the emitted radiance c, particularly with RGB parameters without directional dependence, 16 spherical harmonics (SH), 16 spherical Gaussians (SG), and the combination of 16 SH and SG functions yielding the best results: 9 SH (order 2) and 7 SG. We observe that parameterizations with directional dependence (SH, SG, and SH+SG) significantly improve quality. Furthermore, the parameterization providing the best quantitative results is the combination of SH and SG. This can be justified because SHs are robust for representing low-frequency phenomena, while SGs can adapt to high frequencies.

7. Conclusions, Limitations and Future Work

Limitations and Future Work Although our approach enables high-quality rendering, it has some limitations. Ray casting is computationally intensive, especially with irregularly distributed Gaussians, which results in training times that are not as fast as the current fastest methods, such as Gaussian Splatting. Furthermore, there is still progress in achieving photorealism, for example, by modeling more complex phenomena such as light scattering. Another approach could be to convert our model into a surface model to more easily handle problematic such as scene relighting.

Conclusion Our approach leverages volume ray casting on Gaussian basis functions associated with Spherical Harmonics/Gaussians colorimetric parameters to optimize scene radiance fields and generate state-of-the-art photorealistic renderings, as quantitative and qualitative results demonstrate. Furthermore, we achieve reasonable training times and fast inference times thanks to our implementation of a slab-by-slab ray casting algorithm using a BVH.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, 2022. 7

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19640–19648, 2023. 7

[3] Subrahmanyan Chandrasekhar. Radiative transfer. *Quarterly Journal of the Royal Meteorological Society*, 76(330):498–498, 1950. 2

[4] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting, 2024. 3

[5] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 1

[6] Markus Hadwiger, Ali K. Al-Awami, Johanna Beyer, Marco Agus, and Hanspeter Pfister. Sparseleap: Efficient empty space skipping for large-scale volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):974–983, 2018. 3

[7] Xinlong Ji, Fangneng Zhan, Shijian Lu, Shi-Sheng Huang, and Hua Huang. Mixlight: Borrowing the best of both spherical harmonics and gaussian models, 2024. 5

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), jul 2023. 1, 2, 3, 5, 6, 7, 8

[9] Aaron Knoll, R. Keith Morley, Ingo Wald, Nick Leaf, and Peter Messmer. *Efficient Particle Volume Splatting in a Ray Tracer*, pages 533–541. Apress, Berkeley, CA, 2019. 2

[10] Aaron Knoll, Ingo Wald, Paul Navratil, Anne Bowen, Khairi Reda, Michael E. Papka, and Kelly Gaither. Rbf volume ray casting on multicore and manycore cpus. *Computer Graphics Forum*, 33(3):71–80, 2014. 6

[11] Kubát, J. Radiative transfer in stellar atmospheres. *EAS Publications Series*, 43:1–18, 2010. 4

[12] J. Kulhanek and T. Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18412–18423, Los Alamitos, CA, USA, oct 2023. IEEE Computer Society. 1, 3

[13] H. Liang, T. Wu, P. Hanji, F. Banterle, H. Gao, R. Mantiuk, and C. Öztireli. Perceptual quality assessment of nerf and neural view synthesis methods for front-facing views. *Computer Graphics Forum*, 43(2):e15036, 2024. 8

[14] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 1, 2, 3

[15] Nelson Max and Min Chen. Local and Global Illumination in the Volume Rendering Integral. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 259–274. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2010. 3

[16] 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. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 405–421, Cham, 2020. Springer International Publishing. 1, 2, 7

[17] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), jul 2022. 1, 2, 3, 7

[18] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte carlo methods for volumetric light transport simulation. *Computer Graphics Forum*, 37(2):551–576, 2018. 2

[19] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991. 4

[20] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: a general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4), jul 2010. 6

[21] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 4(43), 2024. 2

[22] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14315–14325, 2021. 3

[23] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. 7

[24] Knut Stamnes, Gary E. Thomas, and Jakob J. Stamnes. *Radiative Transfer in the Atmosphere and Ocean*. Cambridge University Press, 2 edition, 2017. 4

[25] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5459, 2022. 1, 3

[26] Weiwei Sun, Eduard Trulls, Yang-Che Tseng, Sneha Sambandam, Gopal Sharma, Andrea Tagliasacchi, and Kwang Moo Yi. PointNeRF++: A multi-scale, point-based Neural Radiance Field. *arXiv e-prints*, page arXiv:2312.02362, Dec. 2023. 7

[27] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *2022 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 5428–5438, 2022. 1, 3, 7

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

[29] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, New York, NY, USA, 2022. Association for Computing Machinery. 3

[30] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. 2, 3