L_0-Sampler: An L_0 Model Guided Volume Sampling for NeRF

Liangchen Li Juyong Zhang*
University of Science and Technology of China
{vvmno180662@mail., juyong@}ustc.edu.cn

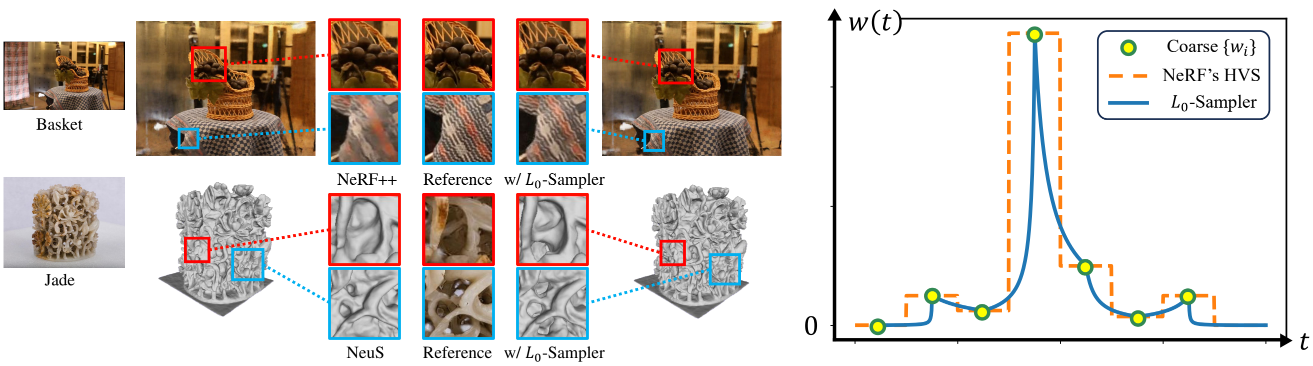


Figure 1. We present L_0-Sampler, an upgrade of the Hierarchical Volume Sampling strategy of NeRF. By testing on different datasets, our proposed L_0-Sampler with different NeRF frameworks can achieve stable performance improvements on rendering and reconstruction tasks with few lines of code modifications and around the same training time. Left: Results comparison between works with original HVS and our L_0-Sampler. Right: Instead of using piecewise constant functions when fitting $w(t)$, we use piecewise exponential functions for interpolation to get a quasi-L_0 $w(t)$, resulting in more concentrated and precise sampling.

Abstract

Since its proposal, Neural Radiance Fields (NeRF) has achieved great success in related tasks, mainly adopting the hierarchical volume sampling (HVS) strategy for volume rendering. However, the HVS of NeRF approximates distributions using piecewise constant functions, which provides a relatively rough estimation. Based on the observation that a well-trained weight function $w(t)$ and the L_0 distance between points and the surface have very high similarity, we propose L_0-Sampler by incorporating the L_0 model into $w(t)$ to guide the sampling process. Specifically, we propose using piecewise exponential functions rather than piecewise constant functions for interpolation, which can not only approximate quasi-L_0 weight distributions along rays quite well but can be easily implemented with a few lines of code change without additional computational burden. Stable performance improvements can be achieved by applying L_0-Sampler to NeRF and related tasks like 3D reconstruction. Code is available at https://ustc3dv.github.io/L0-Sampler/.

*Corresponding Author

1. Introduction

The advent of Neural Radiance Fields (NeRF) [29] has revolutionized the field of inverse rendering, providing powerful solutions for tasks like novel view synthesis, 3D surface reconstruction [55, 61], and dynamic deformation [37, 41]. Volume rendering plays a crucial role in the success of NeRF, as it optimizes the density and color networks to calculate pixel colors. It involves tracing rays through pixels and sampling points along the rays. By leveraging the volume rendering formula, NeRF combines the features of these sampled points to determine the final color. We are aware that in most cases, most sampled points are unoccupied and have little influence on the final result. As a result, the colors obtained through volume rendering mainly rely on points near the surface. As illustrated in Figs. 2a and 2b, providing an accurate geometry to guide the sampling process can significantly improve the training speed and final convergence results. Therefore, a key research problem to further improve volume rendering methods such as NeRF is improving the sampling efficiency and concentrating the sampling points as close to the surface as possible.

NeRF introduces the Hierarchical Volume Sampling (HVS) strategy, inspired by [19], as an efficient approach for sampling near the surface. HVS utilizes volume densities to generate a weight function $w(t)$ and normalize it as a Probability Density Function (PDF) at the coarse stage of each ray. This PDF guides the fine sampling process, leading to improved rendering quality. However, as illustrated in Fig. 1 (right), in the HVS of NeRF, the weight functions are approximated using piecewise constant functions, resulting in a relatively coarse estimation. No matter how accurate the weights of the coarse stage can be, the sampling points at most remain uniformly distributed within a specific interval. Although there has been a lot of works to improve its sampling process since NeRF was proposed, HVS is currently the most stable and versatile and, therefore, the most widely used sampling strategy. Considering the wide use of HVS, its further improvements will benefit volume rendering-based neural rendering and related tasks like 3D reconstruction.

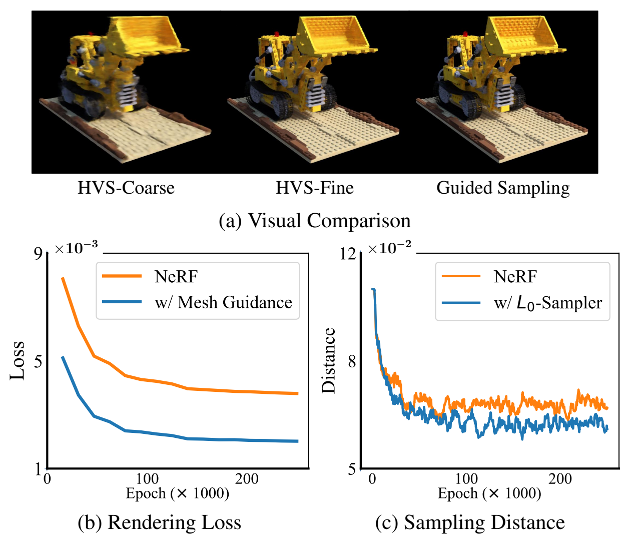


Figure 2. **Importance of Accurate Sampling.** (a) Left: Coarse rendering in HVS. Middle: Fine rendering in HVS. Right: Rendering using a well-trained density network to guide sampling. (b) Rendering loss comparison of NeRF and NeRF with ground truth Lego mesh for accurate sampling. (c) Average distance between sampled points and ground truth Lego mesh during training, showing our accelerated sampling convergence towards real geometry.

In this paper, we propose L_0-*Sampler*, which further improves the sampling process of the HVS method. Our key insight is quite straightforward: when a ray intersects a surface, the volume rendering weight function $w(t)$ of points along the ray are primarily 0, except for very few points around the surface. This behavior is analogous to the L_0 distance between space points and the surface. Therefore, by approximating the weight function to the indicator function, we can make the sampling points approximate the potential object surface as quickly as possible, thus accelerating the training speed and final training results. And our proposed method is entirely different from these sparsity loss term based methods [51, 62] as we directly utilize the L_0 model to guide the selection of sampling points rather than applying a sparsity loss to the density distribution. Our results shown in Tab. 2 also confirm the superiority of our proposed method.

To achieve this target, we propose to construct suitable base functions to interpolate a quasi-L_0 weight function. Through a comprehensive study, we find that piecewise exponential functions shown in Fig. 1 (right) serve as highly suitable base functions. Specifically, they possess the ability to adaptively adjust the gradient within their intervals based on the weights being interpolated, resulting in stable and effective performance across various tasks. As illustrated in Fig. 2c, the utilization of the L_0-Sampler approach brings the sampling points closer to the actual surface.

By applying L_0-Sampler to different NeRF frameworks including NeRF [29], NeRF++ [65], Instant NGP [31], mip-NeRF [4] and NeRF based surface reconstruction NeuS [55], we have observed stable performance improvements, demonstrating its adaptability across diverse datasets and techniques. In addition, one of its particularly important characteristics in practical applications is that its implementation is quite simple and parameter-free. It only requires around ten lines of code to transition from the HVS of NeRF to our method, and each step has a closed-form solution without introducing extra computational overhead. In summary, our main contributions include the following:

- We propose the L_0-Sampler, an enhanced sampling strategy that concentrates sampling by shaping $w(t)$ to approximate the L_0 distance form.
- We analyze the required properties of the interpolation base functions and utilize the piecewise exponential function to interpolate a quasi-L_0 $w(t)$.
- Our L_0-Sampler can stably improve the performance of image rendering and surface reconstruction, and it is parameter-free and can be easily implemented without extra computational overhead.

2. Related Work

Volume Rendering & Surface Rendering. Differentiable rendering mainly includes two types: surface rendering and volume rendering. Surface rendering, exemplified by approaches like DVR [33] and IDR [60], optimizes surfaces based on multi-view images, focusing on radiance determination and often employing implicit gradients. Among them, the signed distance function (SDF) is a popular surface representation, extensively utilized in numerous studies [13, 36, 60]. Volume rendering techniques [25, 29] incorporate both density and color fields, effectively rendering semi-transparent materials but lacking in precise surface definition. This method computes pixel color via a weighted sum along a ray [19], with sampling crucial in weight determination. Prior studies [27] have addressed the complexities of sampling and interpolating density functions in

3D spaces, and [54] fixes quadrature instability in volume rendering. Recent efforts, including [11, 18, 35, 53], have aimed to merge volume and surface rendering to balance rendering quality and geometric accuracy. Our work aligns with these efforts by modifying the weight function $w(t)$ to better highlight surface features.

Neural Radiance Field. The introduction of Neural Radiance Field (NeRF) [29] has significantly impacted view synthesis and depth estimation, inspiring numerous improvements [7, 63] to it. A diverse array of hybrid models have been explored to optimize efficiency, including voxel grids utilized in DVGO [45] and point clouds implemented in Point-NeRF [58]. Similarly, innovative structures like hash grids in Instant NGP [31], relu fields [20], sparse grids in Plenoxels [42], and proposal networks in mip-NeRF 360 [5] are proposed to accelerate training.

Moreover, the adaptability of NeRF stretches far and wide across a variety of applications: representing human figures [12, 15, 39], surface reconstruction [35, 55, 61], dynamic scene modeling [6, 9, 14, 37, 41, 52], deformation tasks [34, 37, 38], and even relighting [44, 69]. It adapts to unbounded scenes [5, 65] with neural networks for background modeling. The proliferation of NeRF methodologies has led to the creation of comprehensive code frameworks like Nerfstudio [48], NeRF-Factory [17], Kaolin-Wisp [47], and NerfAcc [23], which integrate many advanced algorithms.

NeRF Sample Strategy Improvement. The hierarchical volume sampling (HVS) [22] of NeRF has improved the sampling strategy. Subsequent works enhance it from different angles. Efficient ray sampling methods like depth maps and contextual probability distributions [46, 68] optimize the process. Auxiliary networks in papers include [2, 10, 21, 32, 40] boost accuracy and efficiency. Mip-NeRF [4] samples conical frustums to reduce aliasing, while DDNeRF [8] fits Gaussian distributions for precise density representation. Some methods, like light field works [3, 43, 56], sample once per ray without density reliance. Paper [24] proposes automatic integration, and [30] uses specific ray properties for color approximation.

3. Background and Motivation

Given a point $\mathbf{p}$ and a set S, $\mathbf{p}, S \in \mathbb{R}^n$, the L_0 distance between $\mathbf{p}$ and S is defined as follows:

$$d_0(\mathbf{p}, S) = \begin{cases} 1 & \text{if } \mathbf{p} \notin S \\ 0 & \text{if } \mathbf{p} \in S. \end{cases} \tag{1}$$

A distinctive characteristic of this metric is its discontinuous nature, exhibiting an abrupt transition when $\mathbf{p}$ crosses the surface S. Shifting our focus to Neural Radiance Fields (NeRF) [29], NeRF learns to map each point $\mathbf{p}$ and direction vector $\mathbf{d}$ in 3D space to a view-dependent radiance $c(\mathbf{p}, \mathbf{d})$ and a view-independent density $\sigma(\mathbf{p})$. The expected color $C(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, constrained by the bounds t_n and t_f, is computed as:

$$\begin{gathered} C(\mathbf{r}) = \int_{t_n}^{t_f} w(t)\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \\ \text{where } w(t) = \sigma(\mathbf{r}(t)) \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \end{gathered} \tag{2}$$

Here, the weight function $w(t)$ represents the contribution of the point color at $\mathbf{r}(t)$ to the cumulative color of the ray. If a distinct surface S is present, the density $\sigma(\mathbf{p})$ is generally negligible for most points in space, only significantly increasing when approaching the surface. Consequently, it leads $w(t)$ to present the following form:

$$w(t) = \begin{cases} 1 & \text{if } \mathbf{r}(t) \in S \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The function $w(t)$ displays a binary-like behavior that closely resembles the L_0 distance between the point $\mathbf{r}(t)$ and the surface S. We define this correspondence as the $\boldsymbol{L_0}$ **property** of $w(t)$. This property presents the interactions of light with surfaces, aligning well with real-world rendering.

The weight function $w(t)$ is crucial not only for rendering but also for guiding efficient sampling. Observing that points with $w(t) = 0$ have a minor impact on the final rendering, computing values at these points is a redundant operation. Thus, it is more efficient to target sampling in regions close to the surface, where w is much larger. Such an efficient sampling can be achieved by normalizing $w(t)$ into a PDF and using it for inverse transform sampling. However, obtaining a continuous representation of $w(t)$ is impractical. Instead, we turn to a coarse-to-fine estimation strategy. Initially, points $\{\mathbf{r}(t_i)\}$ are sampled uniformly along the ray, and their weights $\{w_i\}$ are predicted using the density outputs $\{\sigma_i\}$ of the coarse network as follows:

$$\alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right), \qquad w_i = \alpha_i \prod_{j=0}^{i-1} \left(1 - \alpha_j\right). \tag{4}$$

Where $\delta_i = t_{i+1} - t_i$ represents the interval length. Subsequently, we extend the weights $\{w_i\}$ to get a function $w(t)$ continuously defined along the entire ray. By ensuring $w(t)$ possesses the desired L_0 property, we can focus most of our sampling points at key locations, enhancing efficiency in the fine sampling stage. A naive solution is to take:

$$w(t) = \begin{cases} 1 & \text{if } t \text{ is near } t^* := \arg\max \{w_i\} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

While this $w(t)$ guarantees the L_0 property and focuses the sampling positions, it may lead to significant errors if the focus deviates from the true surface, which often happens in

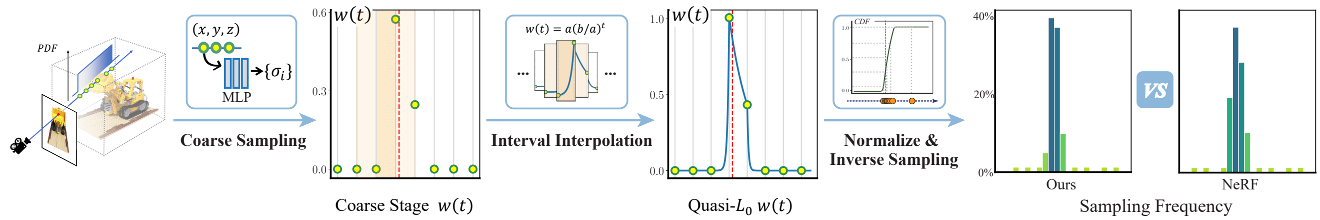


Figure 3. **An Overview of Our L_0-Sampler Pipeline.** The red dashed line represents the surface. During hierarchical volume sampling, we first uniformly sample some points on each ray as NeRF in the coarse stage, and then through piecewise interpolation by interval (*e.g*. Eq. (10)), we fit a quasi-L_0 $w(t)$ resembling an indicator function, which is in line with the L_0 distance between points and surface. After normalization (*e.g*. Eq. (11)), it can be used as a PDF to guide inverse transform sampling (*e.g*. Eq. (12)). The sampling frequency in each interval (right) shows that our method can make the sampling more focused near the surface.

the early stages of training. And as discussed in Sec. 1, the HVS of NeRF extends $\{w_i\}$ into a piecewise constant $w(t)$, uniformly sampling within each interval, resulting in a weak L_0 property. To address this challenge, we adopt piecewise interpolation to obtain a **quasi-**L_0 $w(t)$. This technique balances the optimization of the residual space while preserving the L_0 property to a certain degree. The specifics of this interpolation technique and its role in achieving a quasi-L_0 $w(t)$ will be elaborated in the following section.

4. Method

We now turn to a single ray $\mathbf{o}+t\mathbf{d}$. As discussed above, after obtaining $\{w_i\}$ in the coarse stage, we seek a $w(t)$ with the quasi-L_0 property that satisfies $w(t_i) = w_i$ and enhances sampling efficiency for the fine stage. Unlike NeRF, which uses interval midpoints for $\{w_i\}$, our method interpolates $w(t)$ using both weights at interval endpoints. The integral of $w(t)$ over $[t_i, t_{i+1}]$ satisfies:

$$\int_{t_i}^{t_{i+1}} w(t)dt = (t_{i+1}-t_i)\int_0^1 w((t_{i+1}-t_i)s+t_i)ds. \quad (6)$$

Since $\{t_i\}$ is uniformly sampled, the factor $(t_{i+1} - t_i)$ can be disregarded after normalization. Our analysis, therefore, centers on functions within $[0, 1]$. After being transformed back into $[t_i, t_{i+1}]$ and combined, they collectively establish the comprehensive $w(t)$ along the ray. In Sec. 4.1, we explore the key properties necessary for effective interpolation. Based on that, we will give our solution in Sec. 4.2.

4.1. Interpolation Techniques

Our attention turns to the interval $[t_i, t_{i+1}]$ and its mapping to the normalized interval $[0, 1]$. Here, we introduce the transformed weight function $\hat{w}(s)$, defined as $\hat{w}(s) := w\left((t_{i+1} - t_i)\, s + t_i\right)$, with the boundary values $\hat{w}(0) = a$ and $\hat{w}(1) = b$, constrained by $0 \leq a, b \leq 1$ as a weight. The purpose is to interpolate $\hat{w}(s)$ within this unit interval in a way that closely resembles the L_0 property, *i.e*. achieving a quasi-L_0 behavior that enhances sampling efficiency. First, we will enumerate and discuss the key properties these interpolation functions should possess to meet our optimization requirement.

Property I: Capable of Accurate Interpolation. Formally, $\hat{w}(s)$ should satisfy $\hat{w}(0) = a$ and $\hat{w}(1) = b$.

Property II: Monotonic Within the Interval. This property biases sampling towards the interval end with the greater weight. Considering the unknown location of the point with maximum weight within the interval, we prefer sampling to converge on the endpoints for consistency. Besides, a monotonic $\hat{w}(s)$ ensures that the integrated weight across any interval $[t_i, t_{i+1}]$ is at least as great as the minimum weight at the endpoints, *i.e*.:

$$\int_{t_i}^{t_{i+1}} w(t)dt = \int_0^1 \hat{w}(s)ds \geqslant \min\{a, b\}. \quad (7)$$

Therefore, intervals with higher endpoint weights are more likely to be sampled after normalization, making sampling more concentrated toward them.

Property III: Biased Towards the Larger-weight Side. This property is important in providing our function with the quasi-L_0 property, especially as training progresses. Within any small segment ds, the sampling probability at point s is $(\hat{w}(s)ds) \,/\, \left(\int_0^1 \hat{w}(s)ds\right)$. Assume that $a \leqslant b$, then sampling should be skewed towards $s = 1$, *i.e*. where $\hat{w}(s)$ is higher. To measure this bias, we introduce a weighting function $f(s)$ that increases from 0 to 1 across the interval. A linear weight function, $f(s) = s$, is a straightforward choice, generating the bias metric:

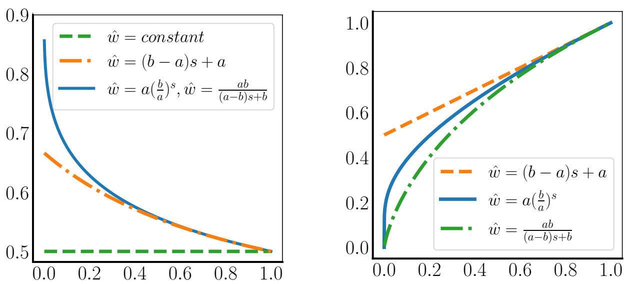
$$bias(\hat{w}) := \frac{\int_0^1 f(s)\hat{w}(s)ds}{\int_0^1 \hat{w}(s)ds} = \frac{\int_0^1 s\cdot\hat{w}(s)ds}{\int_0^1 \hat{w}(s)ds}. \quad (8)$$

This metric, in fact, is the barycenter of the area under the curve $\hat{w}(s)$ over the interval $[0, 1]$. A larger $bias(\hat{w})$ means the $\hat{w}(s)$ has a stronger L_0 property. In that case, functions are typically "steep" in shape near $s = 1$. Notably, the $bias(\hat{w})$ is dependent on endpoint values a and b. For a fixed $\hat{w}(s)$ shape with $b = 1$, $bias(\hat{w})$ becomes a function of a. This dependency is visualized in Fig. 4a. When $a = b = 1$, monotonicity makes $\hat{w}(s) = 1$, centralizing the barycenter at $t = 0.5$. The contrast is most evident when a is small. A large a, b gap implies b is likely near a surface, prompting us to shift the bias towards $s = 1$.

However, an excessive focus on the barycenter can lead to issues since it only represents the spread of sampling points within a specific interval. First, an excessively steep function $\hat{w}(s)$, may results in:

$$\hat{w}(s) \to \min\{w_i, w_{i+1}\}. \tag{9}$$

It can result in an almost uniform distribution. Furthermore, during normalization, the sampling probability in $[t_i, t_{i+1}]$ is proportional to the integral of $w(t)$ over that interval. Therefore, when the difference between a and b is too large, the integral within certain intervals might approximate $\min\{a, b\}$ as depicted in Fig. 4b. This can lead to a lower density of sampling points in these regions. Hence, finding a balance in the steepness of $\hat{w}(s)$ is crucial to ensure the overall sampling distribution is focused as intended.

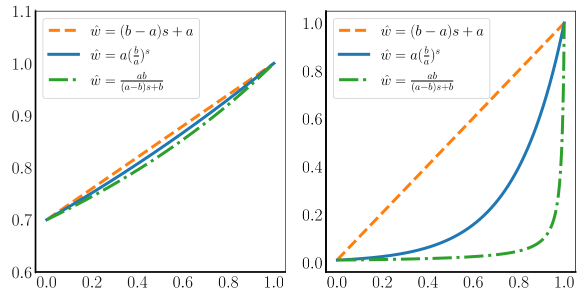


(a) $bias(\hat{w})$ (b) $\int_0^1 \hat{w}(s)ds$

Figure 4. **Function Properties with $\mathbf{b = 1}$.** (a) The $bias$ of some base functions $\hat{w}(s)$. (b) Their integrals over the interval $[0, 1]$. Refer to Fig. 5 for function shapes. Specifically, when a is relatively small, the variations in properties become more evident.

Property IV: Computationally Efficient. The HVS strategy originally used by NeRF is computationally efficient due to the simplicity of piecewise constant functions, which are straightforward to integrate into a PDF and to use for inverse transform sampling. To preserve this efficiency, the cumulative distribution function $\hat{W}(x) := \int_{t_n}^{x} \hat{w}(s)ds$ must have an explicit form, and the equation $\hat{W}(x) = r$ should be easily solvable for $r \geqslant 0$. It requires that $\hat{w}(s)$ remains appropriately simple. Our experiments further indicate that simple functions are sufficient to fulfill this task.

To summarize, when selecting a quasi-L_0 $\hat{w}(s)$, it is required to consider the properties we discussed above. Our pipeline is illustrated in Fig. 3. After determining the interpolation of $\hat{w}(s)$ within each interval, we concatenate these



(a) $a = 0.7, b = 1$ (b) $a = 0.01, b = 1$

Figure 5. **Our Solution.** They can adaptively change their shapes. (a) When a and b are similar, they approximate a linear shape. (b) When a is much smaller than b, they become steep and lead the sampling points towards the interval ends.

functions to form the continuously defined $w(t)$. Then, similar to NeRF, we achieve more precise sampling results by normalizing it to a PDF and applying inverse transform sampling. In fact, the HVS of NeRF is essentially a specific case within our proposed process. It interpolates $w(t)$ using a piecewise constant function and guides sampling with it. Thus, our approach generalizes the original HVS, offering a more universally applicable method.

4.2. Proposed Solution

Following our analysis, the most suitable function we have found is defined as:

$$\hat{w}(s) = a\left(\frac{b}{a}\right)^s. \tag{10}$$

This function is selected due to its monotonic behavior, steep gradient, and simplicity. Notably, its integral over the interval $[0, 1]$ is necessary for normalizing $w(t)$ to get the PDF and is calculated as follows:

$$s(\hat{w}) = \int_0^1 a\left(\frac{b}{a}\right)^s ds = \frac{b-a}{\ln b - \ln a}. \tag{11}$$

The following equation allows us to use inverse transform sampling to find the sampling position x for any residual probability r $(0 \leqslant r \leqslant s(\hat{w}))$ in the interval:

$$r = \int_0^x a\left(\frac{b}{a}\right)^s ds \Rightarrow x = \frac{\ln\left[\frac{r(\ln b - \ln a)}{a} + 1\right]}{\ln b - \ln a}. \tag{12}$$

Moreover, when the values of a and b are relatively close, as shown in Fig. 5a, this suggests the surface here is ambiguous. In these situations, our function behaves more like a linear one, promoting a more uniform sampling within that interval. Conversely, when there is a significant disparity between a and b, exemplified in Fig. 5b, the function curve becomes steeper, resembling an exponential

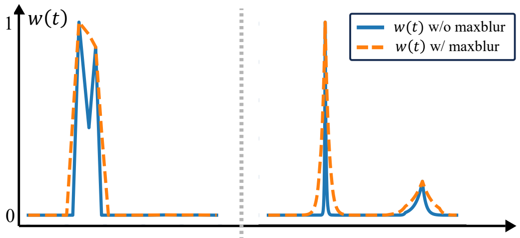


Figure 6. **Effects of Maxblur.** Left: It makes $w(t)$ smoother. Right: By broadening the peak areas of $w(t)$, it increases the probability of sampling within these intervals.

form. This steepness causes its barycenter to shift towards 1, granting it a quasi-L_0 property, as depicted in Fig. 4a. This shift results in more focused sampling in areas with more precise surfaces. Essentially, this represents an adaptive sampling strategy controlled by the ratio b/a.

Additionally, we also consider the piecewise inverse function:

$$\hat{w}(s) = \frac{ab}{(a-b)\,s + b}. \tag{13}$$

The function is named because it is derived from $1/s$. The equations it needs in normalization and inverse transform sampling are:

$$s\,(\hat{w}) = \int_0^1 \hat{w}(s)ds = \frac{ab}{b-a}\,(\ln b - \ln a)\,, \tag{14}$$

$$r = \int_0^x \hat{w}(s)ds \Rightarrow x = \frac{b}{a-b}\left[\exp\left(\frac{r\,(a-b)}{ab}\right) - 1\right]. \tag{15}$$

It exhibits similar properties to the exponential function and can often yield satisfactory results. As illustrated in Fig. 4a, it has the same $bias(\hat{w})$ as $a\,(b/a)^s$, indicating its comparable effects within the interval. However, its performance is less consistent, possibly due to an excessive steepness demonstrated in Fig. 4b, which may affect the sampling probability adversely, as discussed in Sec. 4.1.

Besides, to further refine the weight distribution before sampling, we incorporate the "maxblur" technique from mip-NeRF [4, 66] into our framework:

$$w_i' = \frac{1}{2}\left(\max\left(w_{i-1}, w_i\right) + \max\left(w_i, w_{i+1}\right)\right) + 0.01. \tag{16}$$

This adjustment generates a smoother weight distribution that is closer to reality. We find that this modification works well with our L_0-Sampler by broadening the peak area of $\{w_i\}$, which our steep $w(t)$ then sharpens, achieving a more balanced sampling between intervals (Fig. 6).

5. Experiments

5.1. Experimental Settings

Datasets. We select datasets corresponding to those used in the original works. For our evaluations involving NeRF [29], mip-NeRF [4], and Instant NGP [31], we evaluate our approach using the Blender and Real Forward Facing (LLFF) datasets generated by NeRF [29] and LLFF [28] respectively. In the case of NeRF++, we use scenes from the LF dataset [64], each densely covered by hand-held captured images, with camera parameters recovered via Structure from Motion (SfM). To evaluate the impact on NeuS, we utilize cases from the DTU dataset [16] and BlendedMVS dataset [59], offering a diverse range of materials, appearances, and geometries. The DTU dataset scenes each contain 49 or 64 images with a 1600×1200 resolution, while the BlendedMVS scenes are rendered at 576×768. All scenes in the two datasets are provided with masks.

Metrics. To evaluate the rendered results on novel view synthesis, we use PSNR, SSIM [57] (higher is better for both), and the VGG implementation of LPIPS [67] (lower is better). For geometric results, we employ the Marching Cubes algorithm [26] to extract surfaces and measure the reconstruction quality with the Chamfer distances.

Implementation Details. Our implementation of L_0-Sampler, alongside other comparative experiments, is conducted using PyTorch on a single NVIDIA 3090 GPU. When comparing with others, we keep the training methods the same as those used in previous works, with the only difference being our new sampling technique. Notably, for the specific sampling method of mip-NeRF, which involves conical frustums, we adopt the density of each frustum as the density of its interval midpoint to enable interpolation.

5.2. Comparisons

Quantitative Comparison. We integrate our L_0-Sampler into several works using importance sampling, including NeRF [29], mip-NeRF [4], and the torch version of Instant NGP [31, 49, 50]. Results are shown in Tab. 1. Notably, we consistently improve PSNR across datasets, and the generally lower LPIPS suggests that our method can capture more features of the input images. The outcomes demonstrate the efficacy of our method across various datasets and tasks, further indicating its broad applicability. Although the enhancements may appear modest, the novel perspective from which we update the HVS allows our method to be combined with others, as shown with mip-NeRF and Instant NGP, leading to more accurate and detailed rendering.

Furthermore, our method differs from sparsity loss commonly used in NeRF-related works:

$$\mathcal{L}_{\text{sparsity}} = \beta_s \frac{1}{N}\sum_{k=1}^{N} |1 - \exp(-\delta\sigma_k)|. \tag{17}$$

Methods	Lego		Chair		Drums		Ship		Ficus		Materials		Hotdog		Mic	
	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
NeRF [29]	31.39	0.0400	34.52	0.0283	25.59	0.0741	29.47	0.1409	28.92	0.0416	29.59	0.0432	36.80	0.0298	33.18	0.0272
w/ L_0-Sampler	**31.97**	**0.0346**	**34.92**	**0.0257**	**25.72**	**0.0685**	**29.80**	**0.1342**	**29.21**	**0.0368**	**29.77**	**0.0386**	**37.02**	**0.0278**	**33.61**	**0.0250**
mip-NeRF [4]	33.86	0.0398	33.61	0.0407	24.98	0.0960	28.64	0.1899	31.87	0.0345	30.21	0.0559	36.05	**0.0448**	34.00	**0.0211**
w/ L_0-Sampler	**34.31**	**0.0380**	**33.71**	**0.0405**	**25.12**	**0.0939**	**28.67**	**0.1898**	**32.46**	**0.0302**	**30.34**	**0.0548**	**36.18**	0.0452	**34.02**	0.0214
Instant NGP [31]	32.64	0.0900	32.07	0.1050	23.68	**0.1469**	29.04	**0.1926**	29.51	0.1485	28.02	0.2385	34.68	0.0658	31.57	0.0521
w/ L_0-Sampler	**33.29**	**0.0726**	**33.05**	**0.0852**	**24.05**	0.1616	**29.31**	0.1960	**29.96**	**0.1363**	**28.58**	**0.2218**	**35.66**	**0.0603**	**31.96**	**0.0505**

Table 1. **Quantitative Comparison.** The table compares the performance of various NeRF-based methods to their enhanced versions using our L_0-Sampler on the Blender datasets. Metrics used are PSNR (↑) / LPIPS (↓). We change the sampling strategy in each method into our L_0-Sampler. In NeRF and Instant NGP, we use piecewise exponential functions, while in mip-NeRF we use piecewise inverse functions for interpolation. We observe stable improvements post **almost the same training time** across multiple datasets and tasks.

Method	Lego		Chair		Ficus		Materials	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
NeRF	31.39	0.0400	34.52	0.0283	28.92	0.0416	29.59	0.0432
w/ sparsity loss	31.58	0.0378	34.53	0.0302	28.83	0.0420	29.53	0.0441
w/ L_0-Sampler	**31.97**	**0.0346**	**34.92**	**0.0257**	**29.21**	**0.0368**	**29.77**	**0.0386**

Table 2. **Comparison with Sparsity Loss.** The loss generally improves the rendering results but is not as effective as L_0-Sampler.

Although their idea appears similar to ours, these approaches aim to condense the volume density, while our method concentrates on refining the sampling of points within an already determined density. The comparison is shown in Tab. 2. Here we take $\beta_s = 0.01$ and $\delta = 0.1$ and randomly sample 5000 points in space for loss evaluation. Results show that the sparsity loss brings less improvements in rendering than ours. And our method is parameter-free and needs no additional computations for loss evaluation.

Qualitative Comparison. Fig. 7 provides a qualitative comparison between NeRF, mip-NeRF, and Instant NGP, and their enhanced versions utilizing our L_0-Sampler on the Blender and LLFF datasets. It is evident that our L_0-Sampler helps to capture challenging details, notably highlights, complex textures, and thin structures. And like depth maps in Fig. 8 shows, our sampling captures more geometric details, especially under conditions of sparser sampling points. Furthermore, Fig. 9 depicts the improvement brought about by L_0-Sampler on NeRF++ [65] on Basket case in the LF dataset. Our method enhances the rendering quality of real scenes and reduces rendering artifacts.

Additionally, our more focused sampling results in a more precise capture of geometric details. That makes it beneficial for methods like NeuS [55] that utilize importance sampling in 3D reconstruction. The application of our L_0-Sampler with NeuS shows remarkable improvements in geometry, as depicted in Fig. 10, including correcting unnatural shading-induced pits (DTU 24, 40) and capturing more challenging geometric details (DTU 24 and Fig. 1 left).

Adaptability. It is worth mentioning that our method is still competent in scenes with unclear surfaces, such as clouds and fur. In these cases, the difference in volume densities between adjacent intervals is smaller. Therefore, our PDF will behave like piecewise linear, leading to more uniform sampling, as shown in Fig. 5a. We test with Instant NGP on the bunny_smoke [53] and fox [31] datasets, and the results show that our method can still improve their rendering effects. This proves that our method is highly adaptable.

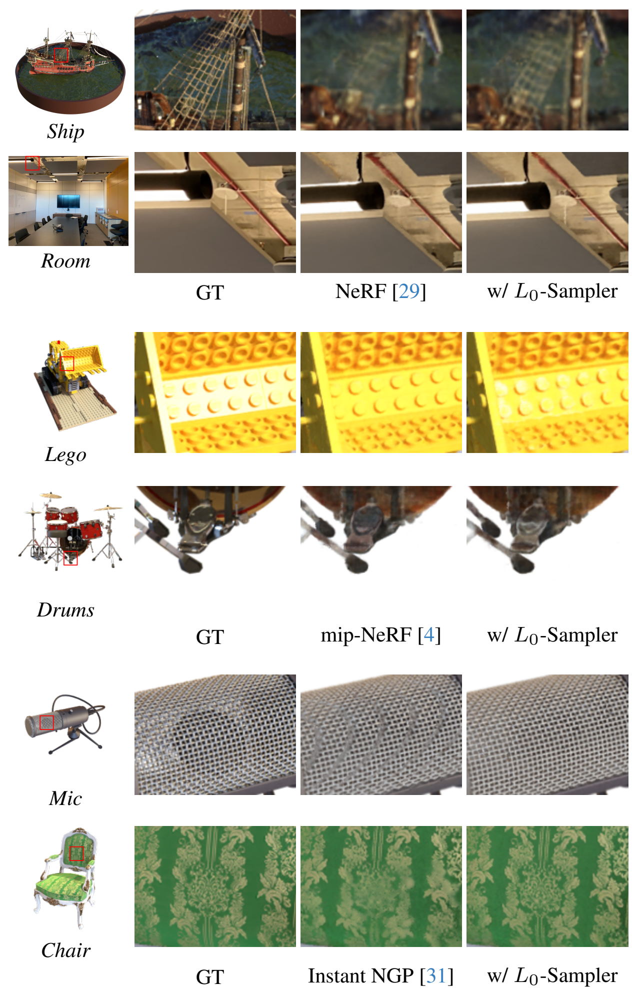
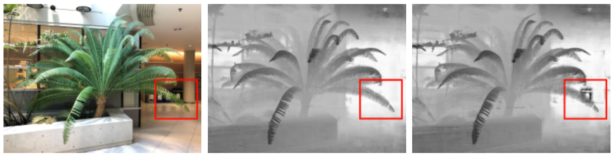
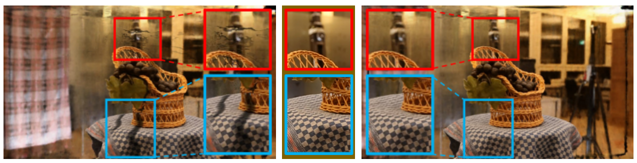


Figure 7. **Qualitative Comparison.** Rendering results of different methods on the Blender and LLFF datasets. Our method shows higher quality in rendering details.



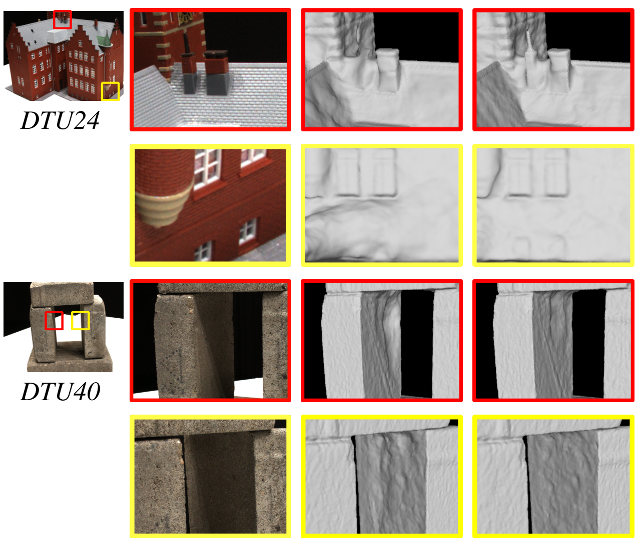
(a) Reference (b) mip-NeRF [4] (c) w/ L_0-Sampler

Figure 8. Depth maps of mip-NeRF with 32 sampling points per ray. Our L_0-Sampler can help it capture more geometry details.



(a) NeRF++ (b) GT (c) w/ L_0-Sampler

Figure 9. L_0-Sampler can alleviate the artifacts in some views.



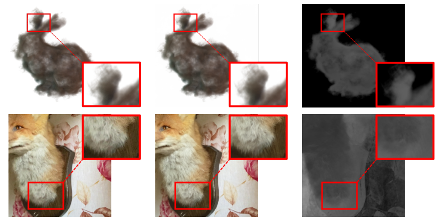
Reference NeuS [55] w/ L_0-Sampler

DTU	24	37	40	55	63	83	122	Mean
NeuS	0.959	0.937	0.545	**0.371**	**1.101**	1.529	**0.495**	0.848
w/ Ours	**0.883**	**0.925**	**0.475**	0.383	1.109	**1.402**	0.497	**0.810**

Figure 10. **Geometry Extraction Quality Comparison.** The data in the table are Chamfer distance values. Our sampling refines NeuS [55] to capture finer details and alleviate shading effects, improving geometry reconstruction accuracy.

5.3. Ablation Study on Interpolation Functions

In our ablation study, we evaluate several interpolation functions alongside those we initially proposed in Sec. 4.2, with the results detailed in Tab. 3. We utilize Instant NGP for training, modifying only the interpolation functions. Among them, the piecewise linear function ($\hat{w}(s) = a + (b - a)s$) is simple yet effective, although it does not focus sampling as much as our method. The piecewise exponential and inverse functions, both part of our proposal,



bunny	INGP	w/ Ours
PSNR	47.61	**48.53**
SSIM	0.993	**0.994**
LPIPS	0.076	**0.074**
fox	INGP	w/ Ours
PSNR	30.44	**30.90**
SSIM	0.882	**0.887**
LPIPS	0.341	**0.333**

GT INGP w/ Ours Depth Comparison

Figure 11. **Results on Smoke and Fur Scenes.** Our method remains effective on surface blur datasets.

Base Functions	Chair			Fern		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Constant	32.07	0.969	0.1050	26.33	0.857	0.1576
Linear	32.86	<u>0.976</u>	<u>0.0866</u>	26.41	0.859	0.1559
Cubic Spline	<u>33.01</u>	0.976	0.0936	26.36	0.858	0.1566
Akima	32.96	0.975	0.0868	26.43	0.861	0.1518
Inverse	32.54	0.975	0.0956	<u>26.54</u>	<u>0.864</u>	**0.1472**
Exponential	**33.05**	**0.977**	**0.0852**	**26.56**	**0.864**	<u>0.1473</u>

Table 3. **Comparison between Interpolation Functions.** The piecewise exponential function delivers consistent improvements, while other functions also enhance the original HVS performance.

show excellent performance in specific cases, particularly the exponential function for its consistent stability.

Additionally, we evaluate the cubic spline and Akima interpolation [1]. Despite the fact that they are widely used, they do not offer the same level of performance in these tasks. This is likely due to the fact that L_0 property is required in real-world weight function fitting rather than good continuity. The choice of function may vary in practice, but our results indicate the potential of our proposed solutions.

6. Conclusion and Discussion

We have proposed L_0-Sampler and applied it to augment the hierarchical volume sampling (HVS), which is the most commonly used sampling strategy in NeRF. Different from previous studies, we adopt a piecewise exponential function to interpolate the weight function $w(t)$ during the sampling process and comprehensively evaluate the effectiveness of this approximation strategy. Its implementation requires only a few lines of code modifications but can produce stable improvements to the NeRF series of works, which has been verified through extensive experiments.

Limitations and Future Work. Although our proposed L_0-Sampler improves performance in the vast majority of cases, not all results are improved. Currently, our method mainly improves the sampling strategy in the fine stage. The idea of how to apply the L_0 model in the coarse stage is also a research direction worth exploring.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 62122071, No. 62272433) and the Youth Innovation Promotion Association CAS (No. 2018495).

References

[1] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17:589–602, 1970. 8

[2] Relja Arandjelović and Andrew Zisserman. Nerf in detail: Learning to sample for view synthesis. *arXiv preprint arXiv:2106.05264*, 2021. 3

[3] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla1, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2, 3, 6, 7, 8

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 5470–5479, 2022. 3

[6] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[8] David Dadon, Ohad Fried, and Yacov Hel-Or. Ddnerf: Depth distribution neural radiance fields. *WACV*, 2023. 3

[9] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 3

[10] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Neusample: Neural sample field for efficient view synthesis. *arXiv preprint arXiv:2111.15552*, 2021. 3

[11] Wanquan Feng, Jin li, Hongrui Cai, Xiaonan Luo, and Juyong Zhang. Neural points: Point cloud representation with neural fields for arbitrary upsampling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[12] Xuan Gao, Chenglai Zhong, Jun Xiang, Yang Hong, Yudong Guo, and Juyong Zhang. Reconstructing personalized semantic facial nerf models from monocular video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 41(6), 2022. 3

[13] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579. 2020. 2

[14] Yudong Guo, Keyu Chen, Sen Liang, Yongjin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[15] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR, 2022. 3

[16] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 6

[17] Yoonwoo Jeong, Seungjoo Shin, and Kibaek Park. nerf-factory: An awesome pytorch nerf library. `https://github.com/kakaobrain/nerf-factory`, 2022. 3

[18] Boyi Jiang, Yang Hong, Hujun Bao, and Juyong Zhang. Selfrecon: Self reconstruction your digital avatar from monocular video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[19] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, page 165–174, New York, NY, USA, 1984. Association for Computing Machinery. 1, 2

[20] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, New York, NY, USA, 2022. Association for Computing Machinery. 3

[21] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. 2022. 3

[22] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 1990. 3

[23] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023. 3

[24] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. 2021. 3

[25] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, 2019. 2

[26] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 6

[27] S.R. Marschner and R.J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings Visualization '94*, pages 100–107, 1994. 2

[28] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 6

[29] 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. *ECCV*, 2020. 1, 2, 3, 6, 7

[30] Nikita Morozov, Denis Rakitin, Oleg Desheulin, Dmitry Vetrov, and Kirill Struminsky. Differentiable rendering with reparameterized volume sampling. *arXiv preprint arXiv:2302.10970*, 2023. 3

[31] 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, 2022. 2, 3, 6, 7

[32] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Computer Graphics Forum*, 2021. 3

[33] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[34] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *International Conference on Computer Vision*, 2021. 3

[35] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[36] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[37] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 1, 3

[38] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 3

[39] Bo Peng, Jun Hu, Jingtao Zhou, Xuan Gao, and Juyong Zhang. Intrinsicngp: Intrinsic coordinate based hash encoding for human nerf. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 3

[40] M. Piala and R. Clark. Terminerf: Ray termination prediction for efficient neural rendering. pages 1106–1114, 2021. 3

[41] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1, 3

[42] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3

[43] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021. 3

[44] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. pages 7491–7500, 2021. 3

[45] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *CVPR*, 2022. 3

[46] Shilei Sun, Ming Liu, Zhongyi Fan, Yuxue Liu, Liquan Dong, and Lingqin Kong. Efficient ray sampling for radiance fields reconstruction. *arXiv:2308.15547*, 2023. 3

[47] Towaki Takikawa, Or Perel, Clement Fuji Tsang, Charles Loop, Joey Litalien, Jonathan Tremblay, Sanja Fidler, and Maria Shugrina. Kaolin wisp: A pytorch library and engine for neural fields research. `https://github.com/NVIDIAGameWorks/kaolin-wisp`, 2022. 3

[48] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 3

[49] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 6

[50] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *arXiv preprint arXiv:2205.14870*, 2022. 6

[51] Jinguang Tong, Sundaram Muthu, Fahira Afzal Maken, Chuong Nguyen, and Hongdong Li. Seeing through the glass: Neural 3d reconstruction of object inside a transparent container. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12555–12564, 2023. 2

[52] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. 2020. 3

[53] Itsuki Ueda, Yoshihiro Fukuhara, Hirokatsu Kataoka, Hiroaki Aizawa, Hidehiko Shishido, and Itaru Kitahara. Neural density-distance fields. In *Proceedings of the European Conference on Computer Vision*, 2022. 3, 7

[54] Mikaela Angelina Uy, George Kiyohiro Nakayama, Guandao Yang, Rahul Krishna Thomas, Leonidas Guibas, and Ke Li. Nerf revisited: Fixing quadrature instability in volume rendering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[55] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 1, 2, 3, 7, 8

[56] Peng Wang, Yuan Liu, Guying Lin, Jiatao Gu, Lingjie Liu, Taku Komura, and Wenping Wang. Progressively-connected light field network for efficient view synthesis. *arXiv preprint arXiv:2207.04465*, 2022. 3

[57] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural

similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 6

[58] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3

[59] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. pages 1787–1796, 2020. 6

[60] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[61] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 1, 3

[62] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[63] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 3

[64] Kaan Yücer, Alexander Sorkine-Hornung, Oliver Wang, and Olga Sorkine-Hornung. Efficient 3d object segmentation from densely sampled light fields with applications to 3d reconstruction. *ACM Trans. Graph.*, 35(3):22:1–22:15, 2016. 6

[65] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 2, 3, 7

[66] Richard Zhang. Making convolutional networks shift-invariant again. *ICML*, 2019. 6

[67] Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. 2018. 6

[68] Wenyuan Zhang, Ruofan Xing, Yunfan Zeng, Yu-Shen Liu, Kanle Shi, and Zhizhong Han. Fast learning radiance fields by shooting much fewer rays. *IEEE Transactions on Image Processing*, 2023. 3

[69] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *TOG 2021 (Proc. SIGGRAPH Asia)*, 2021. 3