Precise Integral in NeRFs: Overcoming the Approximation Errors of Numerical Quadrature

Boyuan Zhang[1], Zhenliang He[1], Meina Kan[1,2], Shiguang Shan[1,2]
[1]Key Lab of AI Safety, Institute of Computing Technology, CAS, China
[2]University of Chinese Academy of Sciences, China
boyuan.zhang@vipl.ict.ac.cn, {hezhenliang,kanmeina,sgshan}@ict.ac.cn

Abstract

Neural Radiance Fields (NeRFs) use neural networks to translate spatial coordinates to corresponding volume density and directional radiance, enabling realistic novel view synthesis through volume rendering. Rendering new viewpoints involves computing volume rendering integrals along rays, usually approximated by numerical quadrature because of lacking closed-form solutions. In this paper, utilizing Taylor expansion, we demonstrate that numerical quadrature causes inevitable approximation error in NeRF integrals due to ignoring the parameter associated with the Lagrange remainder. To mitigate the approximation error, we propose a novel neural field with segment representation as input to implicitly model the remainder parameter. In theory, our proposed method is proven to possess the potential to achieve fully precise rendering integral, as demonstrated by comprehensive experiments on several commonly used datasets with state-of-the-art results.

1. Introduction

Recent progress in neural rendering, especially with neural radiance fields [13], has significantly advanced the capability for creating photorealistic views from new perspectives. NeRF-based methods utilize neural networks to learn a neural field that converts a 3D point coordinates to its corresponding volume density and view-dependent radiance. The vanilla NeRF [25] utilizes a pure MLP structure to implicitly model the neural field. To further improve the rendering fidelity and efficiency, there has been a shift towards methods that incorporate explicitly stored and learnable features alongside fewer MLP layers for the neural field [6, 10, 11, 21, 26, 31, 33, 39].

Upon defining the neural field, NeRFs proceed to the volume rendering process [5, 18], which calculates complex integrals incorporating density and radiance along rays. Finding a closed-form solution for the rendering integral is challenging; therefore, existing NeRF approaches resort to numerical quadrature, an approximate solution for complex integrals, which samples and sums up points along a ray according to the volume density. However, it's important to recognize that numerical quadrature, as an approximate method, has inherent limitations: it could never achieve precise integral with limited sampling points (as illustrated in Fig. 1(a) and Fig. 1(b)). One straightforward approach to reduce the approximation error is to increase the number of sampling points; however, this approach significantly raises the computational costs [38].

Given the inherent limitations of numerical quadrature in achieving precise integrals, it prompts an inquiry: *Is there a feasible approach that could accomplish precise NeRF integral while preserving the computational efficiency akin to numerical quadrature?* In this paper, we derive a precise and achievable formulation for volume rendering integral based on Taylor expansion [1], termed PrecNeRF (Precise NeRF). Through the lens of Taylor expansion, we initially clarify the issue of numerical quadrature: *disregarding the parameter associated with the Lagrange remainder*, which causes the inevitable approximation error in the integral of a segment. In light of this finding, we propose to take into account the remainder parameter that is neglected by numerical quadrature in existing NeRFs. Specifically, while the neural fields in existing NeRFs use a point representation as input, we propose to use a segment representation instead, i.e., the information of the upper and lower limits of the segment integral. In this manner, it can be proved that the remainder parameter can be implicitly modeled by the MLP in the neural field. Ideally, given that the remainder parameter has been properly modeled, our proposed PrecNeRF possesses the potential to achieve fully precise rendering integrals (Fig. 1(c)), which is difficult to be realized in existing NeRFs using numerical quadrature (Fig. 1(a)).

From another perspective, the proposed segment representation provides additional information to eliminate the ambiguity problem (see Sec. 3.4) in existing NeRFs, enhancing the rendering quality.

In summary, we introduce a rigorous and feasible method that eliminates the approximation error of computing rendering integrals in existing NeRF methods. The contributions of this work can be summarized as follows:

- Based on Taylor expansion, we reveal that the numerical quadrature used in existing NeRFs neglects the remainder parameter, which causes the inevitable inaccuracy in computing rendering integrals.
- We propose to change the input of the neural field in existing NeRFs from point representation to segment representation, so that the remainder parameter can be implicitly modeled by the neural network, ideally enabling exact rendering integrals.
- Our proposed method can be painlessly applied to existing NeRFs, and achieves state-of-the-art performance across multiple datasets.

2. Related Work

2.1. Integration Techniques

Closed-form solutions are only possible for some specific integrals. Even when solutions are available, they require complex algorithms to calculate [27–29]. Therefore, for more general functions, using numerical integration [7] to approximate definite integrals is a more feasible method. In numerical integration, two strategies are primarily employed: *numerical quadrature* [41], which is based on deterministic algorithms, and *Monte Carlo methods* [15], which draw on the principles of probability theory.

The core idea of numerical quadrature is to transform a complex integral into a series of simpler summations across all sampled subintervals. As the number of subintervals increases, the accuracy of the estimation typically improves [2]. It could be the summation of areas of rectangles, trapezoids, or parabolas, based on what approximation method is used [7]. The Riemann sum is one of the most basic forms of numerical quadrature. It partitions the integral interval into finite subintervals and uses a constant function to approximate within each subinterval. The constant value is usually the function value of the interval endpoints or midpoints. The summation of all rectangles serves to approximate the overall integral value (Fig. 1(a)). Other methods of numerical quadrature, such as the trapezoidal rule [8] (Fig. 1(b)) and Simpson's Rule [32], can be considered improvements of the Riemann sum. These methods employ different approximation functions (using trapezoids or quadratic polynomials instead of rectangles) to enhance the precision. Adaptive quadrature [12] is another technique that adjusts the lengths of intervals based on the function's behavior. It splits intervals for rapid changes and combines intervals when changes are smoother. By adaptively selecting subintervals, adaptive quadrature strikes a balance between accuracy and efficiency.

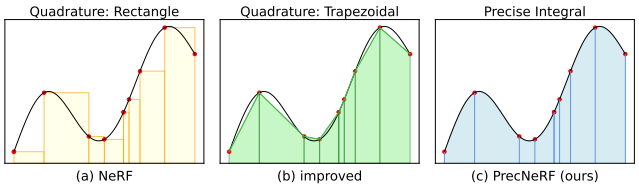

Figure 1. Comparative visualization of numerical quadrature ((a) and (b)) versus precise integral((c)). The black curve represents the function to be integrated and the red points represent the discrete sampling points on this function. (a): NeRF [25] employs adaptive quadrature with rectangular approximation to estimate the integral, resulting in inevitable approximation error. (b): is an improved version of (a) that employs trapezoids instead of rectangles, which often achieves greater estimation accuracy. However, the approximation error still cannot be fully eliminated because the trapezoids fail to perfectly fit the curve. In a word, due to using simple approximation functions such as rectangles and trapezoids, numerical quadrature inherently causes approximation errors. (c): A schematic of our PrecNeRF. Through the lens of Taylor expansion, we discover that the integral value within each segment is a mapping defined by the two endpoints of that segment. Therefore, we opt to leverage powerful neural networks to learn the mapping from segment intervals to their integral values. Theoretically, our method can overcome the approximation errors associated with quadrature, achieving more precise integrals in volume rendering.

Conversely, the Monte Carlo method [15] approximates by randomly selecting points and averaging the function values at these points. The averaged value is then multiplied by the total integral length to get an estimate. The accuracy of the Monte Carlo method is primarily determined by the number of sampling points N (with the standard deviation of estimations being proportional to $1/\sqrt{N}$ [15, 37]). The Monte Carlo method is more effective for high-dimensional integral problems, whereas less efficient for low-dimensional problems like volume rendering integral. Therefore, when estimating the volume rendering integral, adaptive quadrature with rectangular approximation is commonly adopted [25]. Our method also builds on adaptive quadrature but uses complex neural networks to replace the simple rectangular approximation.

2.2. Neural Radiance Field Architectures

To better fit data with high-frequency variation, vanilla NeRF [25] converts the coordinates and ray direction to positional encodings before sending them to the MLP-based neural field. Since the positional encoding does not repre-

sent any physical property of a scene, vanilla NeRF requires a large MLP to store the scene information, which is computationally expensive for training and inference. To improve rendering efficiency, AutoInt [22] proposes automatic integration, a new framework for learning closed-form solutions to rendering integrals. AutoInt enhances efficiency by significantly reducing the number of model queries during inferences but struggles to render high-quality images.

To reduce the reliance on large MLPs, grid-based approaches propose either to store the features of a trained MLP [14, 16, 42] or to directly optimize over specific spatial features [6, 10, 11, 21, 23, 31, 39]. For a query point, its corresponding feature is calculated as an interpolation of the surrounding learned features. Since the learned features capture rich scene information, only a lightweight MLP is required to decode the density and radiance from the interpolated features, which significantly accelerates the rendering speed. However, storing these spatial features requires significant memory and storage capacity. To address this problem, Instant-NGP [26] adaptively stores the spatial features using hash encoding, thereby not only reducing storage requirements but also achieving fast rendering speeds.

To mitigate the blurring and aliasing issues, several methods have chosen to use integrated positional encoding methods. Mip-NeRF [3, 4] first proposes integrated positional encoding that integrates the sinusoidal encoding within a frustum by approximating the frustum by a Gaussian distribution. Exact-NeRF [17] further derives an exact integral of encodings within a tetrahedral frustum. PyNeRF [33] focuses on the aliasing issue of grid-based methods and proposes a simple modification by training model heads at different spatial grid resolutions. DIVER [37] explores to utilize the integration of interpolated features within intervals to achieve deterministic integral, enabling accurate and real-time rendering. PL-NeRF [34] employs the trapezoidal rule when approximating rendering integrals, enabling sharper textures and improved geometry.

In this work, the proposed method is presented as an enhancement plugin for NeRFs, and we demonstrate its effectiveness by building it upon three different frameworks: NeRF [25], NGP [26], and PyNeRF [33].

3. Methods

In Sec. 3.1, we first revisit how existing NeRF approaches compute volume rendering integrals using numerical quadrature. In Sec. 3.2, we then use Taylor expansion to demonstrate the inevitable approximation errors caused by the numerical quadrature. Finally, we introduce our segment neural field in Sec. 3.3 and explain how it prevents the approximation errors in Sec. 3.4.

3.1. Numerical Quadrature in NeRFs

In NeRF [25], a 5D neural radiance field is utilized to represent a scene, capturing both volume density $\sigma(\mathbf{x})$ and directional radiance $c(\mathbf{x}, \mathbf{d})$. Then the color $C(\mathbf{r})$ of ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ passing through the scene is calculated by volume rendering [5, 9, 18, 24]. NeRF [25] separate the ray into segments and calculate the ray color as follows,

$$\mathrm{C}(\mathbf{r}) = \int_{t_0}^{t_n} T(t_0, t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})\mathrm{d}t \tag{1}$$

$$= \sum_{i=0}^{n-1} \left(\int_{t_i}^{t_{i+1}} T(t_0, t_i)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})\mathrm{d}t \right) \tag{2}$$

$$= \sum_{i=0}^{n-1} \left(\prod_{j=0}^{i-1} T(t_j, t_{j+1}) \right) C(t_i, t_{i+1}), \tag{3}$$

where

$$T(t_i, t_{i+1}) = \exp\left(- \int_{t_i}^{t_{i+1}} \sigma(\mathbf{r}(t))\mathrm{d}t \right) \tag{4}$$

and

$$C(t_i, t_{i+1}) = \int_{t_i}^{t_{i+1}} T(t_i, t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})\mathrm{d}t \tag{5}$$

respectively denote the transmittance and the accumulated radiance over the segment $[t_i, t_{i+1}]$. In most NeRF approaches, the density and the radiance are assumed to be constant within each segment, i.e., $\sigma(\mathbf{r}(t)) \approx \sigma(\mathbf{r}(t_i))$ and $c(\mathbf{r}(t), \mathbf{d}) \approx c(\mathbf{r}(t_i), \mathbf{d})$, to allow a closed-form approximation of $T(t_i, t_{i+1})$ and $C(t_i, t_{i+1})$:

$$T(t_i, t_{i+1}) \approx \exp\left(-(t_{i+1} - t_i)\sigma(\mathbf{r}(t_i))\right), \tag{6}$$

$$C(t_i, t_{i+1}) \approx c(\mathbf{r}(t_i), \mathbf{d}) \left(1 - e^{-\sigma(\mathbf{r}(t_i))(t_{i+1} - t_i)}\right). \tag{7}$$

Although the above formulation enables a computable solution for rendering integrals, it inevitably introduces approximation errors.

3.2. Inaccuracy in Numerical Quadrature

In this section, we refer to Taylor expansion to demonstrate the inherent inaccuracy in numerical quadrature. We focus on the transmittance term $T(t_i, t_{i+1})$ in Eq. (4), while the accumulated radiance term $C(t_i, t_{i+1})$ in Eq. (5) has a similar but more complex analysis and we leave it in the supplementary material.

For the transmittance

$$T(t_i, t) = \exp\left(- \int_{t_i}^{t} \sigma(\mathbf{r}(s))\mathrm{d}s \right), \tag{8}$$

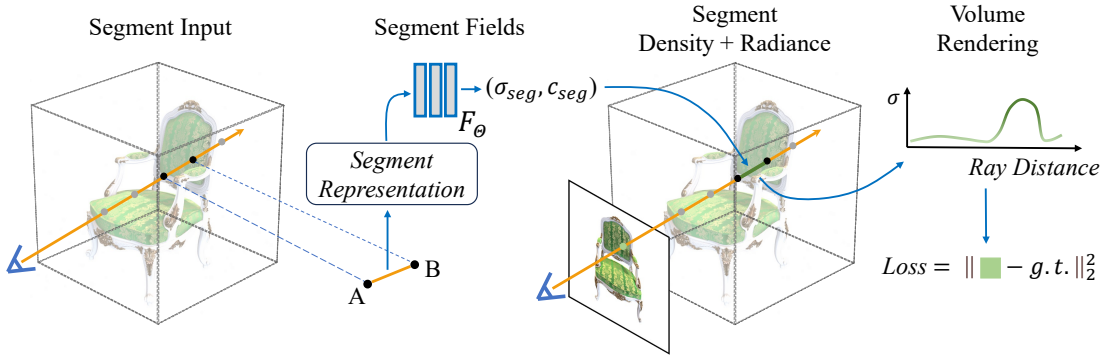

Figure 2. Overview of our PrecNeRF pipeline. For a segment $\overline{\text{AB}}$ on a ray, we compute its representation by concatenating the encoding of its two endpoints, where we can employ sinusoidal encoding [25] or hash encoding [26]. This composite representation is subsequently fed into an MLP network to obtain the integrated information of density and radiance over that segment, i.e., we learn a segment density field and a segment radiance field. Further, the segment density and segment radiance are incorporated into the volume rendering process.

let $f(t)$ denote its exponential component, i.e.,

$$f(t) = -\int_{t_i}^{t} \sigma(\mathbf{r}(s))\mathrm{d}s, \tag{9}$$

and we can derive that

$$f(t_i) = 0, \quad f^{(1)}(t) = -\sigma(\mathbf{r}(t)). \tag{10}$$

By performing 0^{th}-order Taylor expansion to $f(t)$ at t_i with Lagrange remainder term, we obtain

$$f(t) = f(t_i) + f^{(1)}(t_i + \theta(t - t_i))(t - t_i) \tag{11}$$

$$\overset{(10)}{=} -(t - t_i)\sigma\left(\mathbf{r}(t_i + \theta(t - t_i))\right), \tag{12}$$

where $\theta \in [0, 1]$ is the parameter associated with the Lagrange remainder. According to Eq. (8) and Eq. (12), we obtain a precise calculation of $T(t_i, t_{i+1})$ as follows,

$$T(t_i, t_{i+1}) = \exp(-(t_{i+1} - t_i)\sigma(\mathbf{r}(t_i + \underline{\theta(t_{i+1} - t_i)}))). \tag{13}$$

For a clear comparison, Eq. (6) is reproduced below,

$$T(t_i, t_{i+1}) \approx \exp\left(-(t_{i+1} - t_i)\sigma(\mathbf{r}(t_i))\right). \tag{14}$$

Eq. (13) and Eq. (14) respectively show the precise and approximative manner (i.e., numerical quadrature) to calculate $T(t_i, t_{i+1})$. As can be seen, the numerical quadrature in Eq. (14) neglects the underlined $\theta(t_{i+1} - t_i)$ term in Eq. (13), therefore inherently causing an approximation error. From another perspective, $\sigma(\mathbf{r}(t_i + \theta(t_{i+1} - t_i)))$ in Eq. (13) can be defined as a binary function by the two endpoints of the segment, i.e., $\sigma(\mathbf{r}(t_i + \theta(t_{i+1} - t_i))) := \sigma_{seg}(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))$. Apparently, this binary function $\sigma_{seg}(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))$ cannot be precisely modeled by the unary function $\sigma(\mathbf{r}(t_i))$ in numerical quadrature in Eq. (14).

3.3. Precise NeRFs with Segment Neural Fields

In this section, we propose segment neural fields to solve the aforementioned approximation error encountered in numerical quadrature. Specifically, we respectively introduce the segment density field and the segment radiance field to achieve precise calculation of the transmittance and the accumulated radiance. Further, the segment density field and the segment radiance field are incorporated into the volume rendering process, as illustrated in Fig. 2.

Segment Density Field As can be seen from Eq. (13) in Sec. 3.2, figuring out $\sigma(\mathbf{r}(t_i + \theta(t_{i+1} - t_i)))$ is necessary for the precise calculation of the transmittance $T(t_i, t_{i+1})$. However, it is challenging to directly estimate the Lagrange parameter θ; instead, we choose to implicitly model this parameter with a neural density field defined by segments.

As mentioned in Sec. 3.2, $\sigma(\mathbf{r}(t_i + \theta(t_{i+1} - t_i)))$ can be defined as a binary function by the two endpoints of the segment $[t_i, t_{i+1}]$, i.e., $\sigma(\mathbf{r}(t_i + \theta(t_{i+1} - t_i))) := \sigma_{seg}(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))$. By substituting this binary function into Eq. (13), the transmittance becomes

$$T(t_i, t_{i+1}) = \exp\left(-(t_{i+1} - t_i)\sigma_{seg}\left(\mathbf{r}(t_i), \mathbf{r}(t_{i+1})\right)\right). \tag{15}$$

Based on this formulation, we employ a neural network to model σ_{seg} as follows,

$$\sigma_{seg}\left(\mathbf{r}(t_i), \mathbf{r}(t_{i+1})\right) = \mathrm{MLP}_\sigma([\mathrm{Enc}(\mathbf{r}(t_i)), \mathrm{Enc}(\mathbf{r}(t_{i+1}))], \tag{16}$$

where MLP_σ denotes multilayer perceptron, and Enc denotes encoding functions such as sinusoidal encoding [25] or hash encoding [26]. In this manner, we fix the problem of ignoring the Lagrange parameter in numerical quadrature (Sec. 3.2), thus enabling precise rendering integrals.

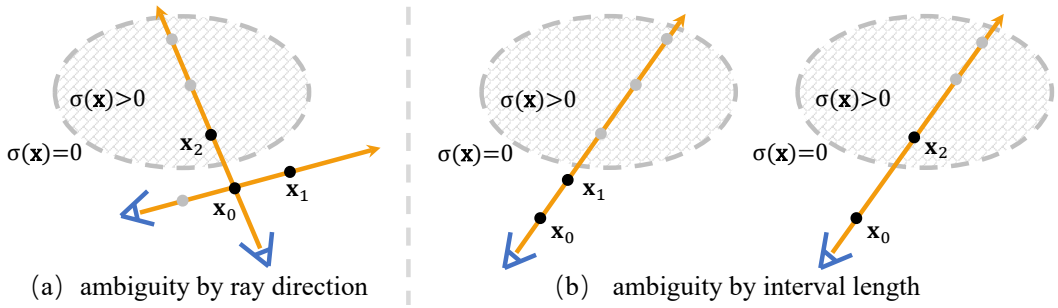

Figure 3. Illustration of the ambiguity in learning a NeRF with numerical quadrature. The learning process tends to optimize the density value of the segment starting point (e.g., $\mathbf{x}_0$ here) towards the average density over that segment. However, $\mathbf{x}_0$ can be the starting point of multiple segments during the training process, e.g., $\overline{\mathbf{x}_0\mathbf{x}_1}$ and $\overline{\mathbf{x}_0\mathbf{x}_2}$, of which the average densities are likely to be distinct. That is to say, the target to learn the density of $\mathbf{x}_0$ is ambiguous.

Segment Radiance Field By performing 1st Taylor expansion with Lagrange remainder at t_i to the accumulated radiance in Eq. (5), we have

$$C(t_i, t_{i+1}) = \underline{\left(\sigma(\mathbf{r}(t_i))c(\mathbf{r}(t_i), \mathbf{d}) + \frac{C^{(2)}(\eta)}{2!}(t_{i+1} - t_i)\right)}(t_{i+1} - t_i), \tag{17}$$

where $\eta \in [t_i, t_{i+1}]$. Similar to the analysis of the segment density field, the underlined term in Eq. (17) can be also defined as a binary function, which we design as

$$C(t_i, t_{i+1}) = \underline{\sigma_{seg}(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))c_{seg}(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))}(t_{i+1} - t_i), \tag{18}$$

where σ_{seg} is the reuse of the segment density field in Eq. (16), and c_{seg} is the segment radiance field with the purpose of learning the color information of a segment. Comparing Eq. (18) to Eq. (17), it can be seen that we utilize these two segment fields to capture the information of the Lagrange remainder, aiming at a precise calculation of the accumulated radiance $C(t_i, t_{i+1})$. Similar to σ_{seg}, c_{seg} is modeled by neural network as follows,

$$c_{seg}\left(\mathbf{r}(t_i), \mathbf{r}(t_{i+1})\right)) = \text{MLP}_c([\text{Enc}(\mathbf{r}(t_i)), \text{Enc}(\mathbf{r}(t_{i+1}))]. \tag{19}$$

NeRFs with Segment Fields Upon defining the segment density field and the segment radiance field, they can be painlessly applied to existing NeRFs by directly replacing the original density and radiance field.

3.4. Discussion About the Ambiguity Issue

In this section, we intuitively illustrate the ambiguity issue caused by numerical quadrature in existing NeRFs for volume rendering and explain why our proposed method can avoid this issue.

During training, according to Sec. 3.1, NeRFs may sample any segment $\overline{\mathbf{x}_i\mathbf{x}_j}$ for the calculation of the transmittance and the cumulative radiance (Eq. (6) and Eq. (7)). Therefore, it is possible to sample two segments with the same starting point such as $\overline{\mathbf{x}_0\mathbf{x}_1}$ and $\overline{\mathbf{x}_0\mathbf{x}_2}$ in Fig. 3. In this case, take the transmittance as an example, $T(\mathbf{x}_0, \mathbf{x}_1)$ and $T(\mathbf{x}_0, \mathbf{x}_2)$ are estimated as follows,

$$T(\mathbf{x}_0, \mathbf{x}_1) \approx \exp\left(-\Delta(\mathbf{x}_0, \mathbf{x}_1)\sigma(\mathbf{x}_0)\right), \tag{20}$$
$$T(\mathbf{x}_0, \mathbf{x}_2) \approx \exp\left(-\Delta(\mathbf{x}_0, \mathbf{x}_2)\sigma(\mathbf{x}_0)\right), \tag{21}$$

hence

$$\sigma(\mathbf{x}_0) \approx -\frac{\ln T(\mathbf{x}_0, \mathbf{x}_1)}{\Delta(\mathbf{x}_0, \mathbf{x}_1)}, \sigma(\mathbf{x}_0) \approx -\frac{\ln T(\mathbf{x}_0, \mathbf{x}_2)}{\Delta(\mathbf{x}_0, \mathbf{x}_2)}, \tag{22}$$

where $\Delta(\mathbf{x}_i, \mathbf{x}_j)$ denotes the length of the segment $\overline{\mathbf{x}_i\mathbf{x}_j}$, and $\sigma(\mathbf{x}_0)$ is the predicted density at point $\mathbf{x}_0$. As shown in Eq. (22), $\sigma(\mathbf{x}_0)$ tends to converge towards two distinct values, indicating an ambiguity, which leads to imprecise estimates of volume density and thus transmittance. As a result, incorrect density estimates make it challenging to reconstruct scene geometry, while incorrect estimates of transmittance lead to brighter or darker rendered colors.

In contrast, our method resolves the ambiguity issue by formulating the neural field as a function of both endpoints of an input segment, as shown in Eq. (15). Specifically, it can be derived from Eq. (15) that our proposed segment density can be calculated as follows,

$$\sigma_{seg}(\mathbf{x}_0, \mathbf{x}_1) = -\frac{\ln T(\mathbf{x}_0, \mathbf{x}_1)}{\Delta(\mathbf{x}_0, \mathbf{x}_1)}, \sigma_{seg}(\mathbf{x}_0, \mathbf{x}_2) = -\frac{\ln T(\mathbf{x}_0, \mathbf{x}_2)}{\Delta(\mathbf{x}_0, \mathbf{x}_2)}. \tag{23}$$

As can be seen, compared to the conflicting results of numerical quadrature in Eq. (22), our results in Eq. (23) have no conflicts, i.e., there is no ambiguity in our formulation.

4. Experiments

In this section, We first detail the datasets and evaluation metrics in Sec. 4.1 and implementation details in Sec. 4.2. We then compare our PrecNeRF's performance against baseline approaches that use adaptive quadrature with rectangular approximation in Sec. 4.3. Observations indicate consistent improvements (see Tab. 1, Tab. 2) and state-of-the-art rendering results (see Fig. 4). We also compare our PrecNeRFs with other approximation methods (see Tab. 3). Leveraging the potent fitting capabilities of neural networks, our methods significantly outperform basic functions like trapezoids and parabolas. Finally, in Sec. 4.4 we validate the robustness of our segment representation, showing superior performance with fewer sampling points compared to numerical quadrature (see Tab. 5 and Fig. 5).

4.1. Datasets and Metrics

To ensure a comprehensive evaluation of our methods, we select three public datasets including real-world and synthetic scenes. The NSVF [23] and synthetic-NeRF [25] datasets both include 100 training, 100 validation, and 200 testing images each, all at a resolution of 800x800, featuring eight synthetic models of common objects. The TanksTemples [19] dataset consists of real-world images from five scenes, with each scene containing approximately 100 to 300 training images at a resolution of 1920x1080. The train-test ratio for TanksTemples is approximately 7. We employ three commonly used evaluation metrics, including the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [36], and Learned Perceptual Image Patch Similarity (LPIPS) [43]. For LPIPS, we employ both VGG [30] and AlexNet [20] as backbones to ensure a comprehensive evaluation.

4.2. Implementation Details

In our work, NeRF [25], NGP [26], and PyNeRF [33] are chosen as baseline methods. Adhering mostly to the original setups, we only differed in some adjustments. For PrecNeRF, we concatenate the sinusoidal encodings of both endpoints and double the number of neurons in the first MLP layer. We conduct one million training iterations with 4096 rays each. The learning rate starts at 5×10^{-4} and decay exponentially to 5×10^{-5} during training. For training stability, we add a 2500-iterations warm-up phase which raises the learning rate from 5×10^{-6} to the initial rate using a cosine scheme, same as Mipnerf [3]. For PrecNGP, the hashed endpoint features are concatenated and the neurons in the first MLP layer are doubled same as PrecNeRF. We train PrecNGP for 50,000 iterations with dynamic batch size adjustments starting from 1024. The learning rate linearly climbs to 1×10^{-2} in the first 100 iterations, then decreases by 33% at the 1/3 and 9/10 of the whole training process. For PrecPyNeRF, we strictly follow the original paper's setup and handle the concatenation of features similar to PrecNGP. We empirically find that $(1 - e^{-\sigma_{seg}(t_{i+1}-t_i)})$ performs better compared to term $(\sigma_{seg} \times (t_{i+1} - t_i))$ in Eq. (18), so we use the former one in practice.

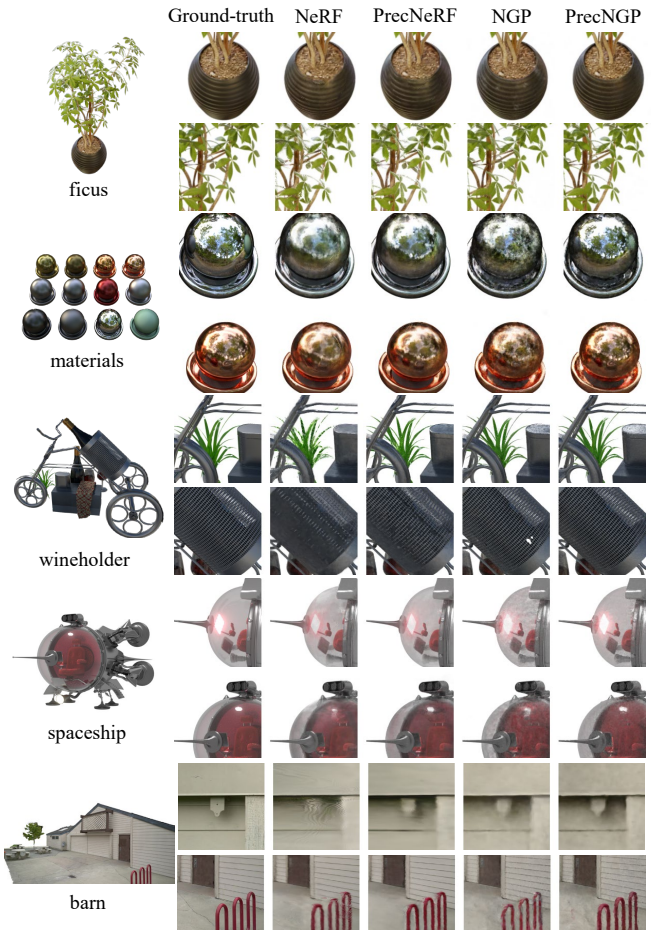

Figure 4. We compare the rendering results of NGP [26] and NeRF [25] with our enhanced PrecNeRF and PrecNGP. By using segment neural fields, we significantly improve the reconstructed scene geometry and the clarity of textures.

4.3. Comparisons

In this section, we first compare our methods (PrecNeRF, PrecNGP, and PrecPyNeRF) with their counterparts (NeRF [25], NGP [26], and PyNeRF [33]) which use adaptive quadrature with rectangular approximation. We then compare PrecNGP with variants of NGP with more complex approximation functions other than rectangles.

PrecNeRFs versus Numerical Quadrature After refining the numerical quadrature into the precise integral, the average test PSNR shows an approximate increase of 1.0 dB for synthetic scenes and 0.4 dB for real-world scenes, as detailed in Tab. 1. Visual comparisons in Fig. 4 high-

Method	Synthetic-NeRF [25] PSNR↑	SSIM↑	LPIPS_v↓	LPIPS_a↓	NSVF [23] PSNR↑	SSIM↑	LPIPS_v↓	LPIPS_a↓	TanksTemples [19] PSNR↑	SSIM↑	LPIPS_v↓	LPIPS_a↓
NSVF [23]	31.75	0.953	-	0.047	35.18	0.979	-	0.015	28.48	0.901	-	0.155
Plenoxels [11]	31.71	0.958	0.049	-	-	-	-	-	27.43	0.906	0.162	-
DVGO [31]	31.95	0.957	-	0.035	35.08	0.975	-	0.019	28.41	0.911	0.155	-
Mipnerf [3]	33.09	0.961	0.043	-	-	-	-	-	-	-	-	-
DIVER [37]	32.32	0.960	-	0.032	-	-	-	-	28.18	0.912	-	0.116
PL-NeRF [34]	31.10	0.948	-	0.044	-	-	-	-	-	-	-	-
TensoRF [6]	33.14	0.963	0.047	0.027	36.52	0.982	0.026	0.012	28.56	0.920	0.140	0.125
NeRF [25]	31.01	0.947	0.081	-	-	-	-	-	-	-	-	-
NeRF* [25]	31.87	0.957	0.056	0.032	34.85	0.974	0.034	0.018	27.46	0.901	0.164	0.136
PrecNeRF (ours)	32.77	0.961	0.050	0.029	36.06	0.979	0.027	0.013	27.83	0.906	0.155	0.124
NGP [26]	33.18	-	-	-	-	-	-	-	-	-	-	-
NGP* [25]	33.36	0.963	0.050	0.027	35.93	0.980	0.026	0.012	28.78	0.918	0.136	0.109
PrecNGP (ours)	34.25	0.967	0.045	0.023	36.86	0.982	0.023	0.010	29.23	0.917	0.132	0.104

Table 1. Quantitive comparisons of our PrecNeRFs against baseline methods and previous methods. For existing methods, scores are sourced directly from their papers whenever possible. The superscript* denotes our replication of the original method. Averaged scores are reported across different scenes. LPIPS_v and LPIPS_a refer to LPIPS_vgg and LPIPS_alex respectively. Higher PSNR↑ and SSIM↑ values are better, while a lower LPIPS↓ is preferred.

	PSNR↑				SSIM↑				LPIPS_alex↓			
	Full	$1/2_{res}$	$1/4_{res}$	$1/8_{res}$	Full	$1/2_{res}$	$1/4_{res}$	$1/8_{res}$	Full	$1/2_{res}$	$1/4_{res}$	$1/8_{res}$
PyNeRF*	32.96	34.74	35.64	35.76	0.963	0.975	0.982	0.987	0.032	0.014	0.008	0.006
PrecPyNeRF (ours)	33.35	35.24	36.32	36.30	0.965	0.977	0.984	0.988	0.030	0.013	0.007	0.005

Table 2. Comparison of PyNeRF [33] and PrecPyNeRF on multi-scale Synthetic-NeRF dataset [33], averaged scores are reported.

light that our method is superior in reconstructing intricate scene geometries and preserving the clarity of textures. The consistent improvements underscore the significance of the precise integral and the efficacy of our segment neural fields. We also compare PrecNeRFs with existing grid-based methods [6, 11, 23, 31] and deterministic integration methods [37] in Tab. 1, our approach leads by a certain margin and achieve state-of-the-art rendering results. Our methods also work well with anti-aliased methods like PyNeRF [33]. We compare our PrecPyNeRF with PyNeRF on the multi-scale synthetic-NeRF dataset [33], which is a challenging multiscale variant of the original synthetic-NeRF dataset. As illustrated in Tab. 2, our method demonstrates its effectiveness by enhancing rendering accuracy across all tested resolutions.

PrecNGP versus Variants of Numerical Quadrature We also compare our PrecNGP with more numerical quadrature variants (e.g. using trapezoids and parabolas other than rectangles to approximate).

Due to the double integral of the volume rendering integral (Eq. (3)), it is still challenging to deduce a closed-form solution if we approximate the density function and radiance function simultaneously. Therefore, we simplify the calculation by assuming a constant radiance within each interval and only approximate the density function. From the quantitative comparisons shown in Tab. 3, we can see that by leveraging the fitting abilities of neural networks, our method surpasses quadrature variants by a large margin.

Method	PSNR↑	SSIM↑	LPIPS_v↓	LPIPS_a↓
Rectangles	33.36	0.963	0.050	0.027
Trapezoids	33.48	0.963	0.049	0.027
Parabolas	33.52	0.964	0.049	0.026
Ours	34.25	0.967	0.045	0.023

Table 3. We compare our precise modeling with classical approximation functions used in numerical quadrature. Due to the complexities of evaluating the double integral of the volume rendering integral, we simplify the process by approximating only the density function while assuming a constant color within each interval.

	Coarse		Fine	
	NeRF	PrecNeRF	NeRF	PrecNeRF
depth - PSNR ↑	19.44	26.96	29.30	32.87
depth - MSE ↓	1.750	0.482	0.415	0.071

Table 4. For the synthetic-NeRF dataset, we render depth maps on the test views from both NeRF and PrecNeRF. We then calculate the PSNR and MSE metrics against the ground truth depth maps. PrecNeRF extracts better geometry. Coarse and Fine correspond to the rendering results from the hierarchical sampling phases.

Comparisons of the Rendered Depth Maps We can divide a ray into multiple segments and identify the segment where the CDF of the ray termination distribution reaches 0.5. The midpoint of that segment can be regarded as the intersection of the light ray and the object surface, i.e., the depth location of that ray. Using this information, we can render depth maps from segment radiance fields. Tab. 4

Step size	PSNR↑		SSIM↑		$\text{LPIPS}_{\text{vgg}}$↓		$\text{LPIPS}_{\text{alex}}$↓	
	NGP*	PrecNGP	NGP*	PrecNGP	NGP*	PrecNGP	NGP*	PrecNGP
0.005	33.36	34.25	0.963	0.967	0.050	0.045	0.027	0.023
0.010	32.12	33.85	0.956	0.966	0.058	0.047	0.033	0.025
0.020	30.07	32.53	0.942	0.959	0.078	0.057	0.048	0.032
0.040	27.91	30.62	0.919	0.945	0.109	0.078	0.074	0.046
0.080	25.42	28.21	0.887	0.920	0.146	0.108	0.111	0.070

Table 5. We compare numerical quadrature and precise integral under different render step sizes at training time. The asterisk superscript (*) denotes our replication of the original method. As the render step size increases (the number of samples decreases), the lead of the precise integral over the numerical quadrature becomes larger.

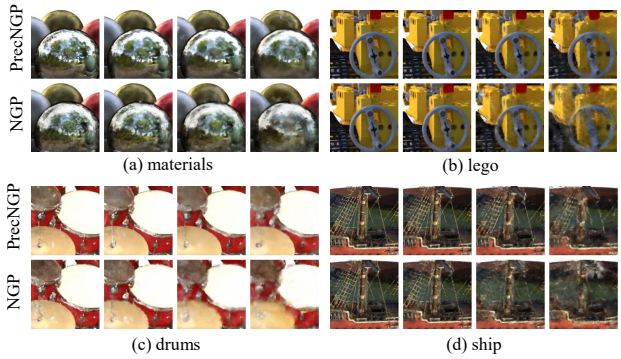

Figure 5. We compare the rendering results of NGP [26] with our PrecNGP when training under limited sampling points. The number of sampling points decreases from left to right whereas the length of the integral interval as well as the approximation ambiguity increases. Compared to NGP, our PrecNGP effectively handles ambiguity, preserving both thin geometries and complex textures (e.g. the non-Lambertian reflections in (a); the circular edges in (b) and (c); the rigging of the ship in (d)).

shows comparisons of the precision of depth maps rendered from NeRF and PrecNeRF. As can be seen, our PrecNeRF can improve the precision of the extracted geometry. Notably, in the coarse sampling phase, due to the fewer samples per ray, the advantage of PrecNeRF in rendering accurate depth maps compared to NeRF is more evident.

4.4. Robustness under Limited Sampling Points

In this section, we show that our precise integral method effectively addresses the ambiguity issue discussed in Sec. 3.4. To make comparisons more intuitive, we reduce the number of sampling intervals when training NGP and PrecNGP on the synthetic-NeRF dataset. The NGP method [26] employs a fixed rendering step size for sampling points along rays, concluding the process once the transmittance falls below a specified threshold. Consequently, by doubling the rendering step size multiple times, we reduce the number of sampling points, and we document the rendering results at each time, see Tab. 5. A larger rendering step size not only diminishes the samples per ray but also significantly amplifies the ambiguity of the training process. Results in Tab. 5 show that the performance of numerical quadrature deteriorates more swiftly compared to our precise integral method under increasing ambiguity. This is because our method utilizes the information of the segment rather than a single point to mitigate the training ambiguity. In Fig. 5 a vivid comparison between the two is presented. It can be observed that, even under severe ambiguity, PrecNGP succeeds in accurately reconstructing the scene geometry and detailed textures, while NGP tends to yield blurrier rendering results.

5. Conclusion

In this paper, we analyze the approximation error caused by the numerical quadrature for the integrals in volume rendering under the perspective of Taylor expansion. According to this analysis, we propose PrecNeRF with a segment density field and a segment radiance field, which can be proven to overcome the approximation errors of numerical quadrature in learning NeRFs. Furthermore, comprehensive experiments demonstrate that our proposed method improves several NeRFs on commonly used datasets with state-of-the-art results.

Currently, our network does not strictly adhere to the multiplicative property of the transmittance function (i.e., $T(\mathbf{x}_0, \mathbf{x}_2) = T(\mathbf{x}_0, \mathbf{x}_1) * T(\mathbf{x}_1, \mathbf{x}_2)$) due to the absence of an explicit constraint during training. Nevertheless, this concept presents a compelling opportunity to enhance our model. First, we can enforce this property as a constraint during training to improve the geometric characteristics. Furthermore, it might also help reduce the number of segments during inference, thereby enhancing the efficiency of the model. Additionally, current methods usually utilize point fields (i.e., SDF) to extract geometry [35, 40]. Consequently, these methods cannot be directly applied to our segment fields for extracting scene geometry. Therefore, it is imperative for future work to investigate the impact of the multiplicative property and to develop methods for converting segment fields into point fields.

Acknowledgement This work is partially supported by the National Natural Science Foundation of China (No. 62406311 and No. 62122074), the Postdoctoral Fellowship Program of CPSF (No. GZB20230774), and the Innovation Funding of ICT, CAS (No. E361020).

References

[1] Tom M Apostol. One-variable calculus, with an introduction to linear algebra. (No Title), 1967. 1

[2] Kendall Atkinson. An introduction to numerical analysis. John wiley & sons, 1991. 2

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5855–5864, 2021. 3, 6, 7

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

[5] Subrahmanyan Chandrasekhar. Radiative transfer. courier corporation, 2013. 1, 3

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In European Conference on Computer Vision, pages 333–350. Springer, 2022. 1, 3, 7

[7] Philip J Davis and Philip Rabinowitz. Methods of numerical integration. Courier Corporation, 2007. 2

[8] Sever S Dragomir, Pietro Cerone, and Anthony Sofo. Some remarks on the trapezoid rule in numerical integration. RGMIA research report collection, 2(5), 1999. 2

[9] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. ACM Siggraph Computer Graphics, 22(4):65–74, 1988. 3

[10] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12479–12488, 2023. 1, 3

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5501–5510, 2022. 1, 3, 7

[12] Walter Gander and Walter Gautschi. Adaptive quadrature—revisited. BIT Numerical Mathematics, 40:84–101, 2000. 2

[13] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. arXiv preprint arXiv:2210.00379, 2022. 1

[14] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 14346–14355, 2021. 3

[15] John Geweke. Monte carlo simulation and numerical integration. Handbook of computational economics, 1:731–800, 1996. 2

[16] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5875–5884, 2021. 3

[17] Brian KS Isaac-Medina, Chris G Willcocks, and Toby P Breckon. Exact-nerf: An exploration of a precise volumetric parameterization for neural radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 66–75, 2023. 3

[18] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. ACM SIGGRAPH computer graphics, 18(3):165–174, 1984. 1, 3

[19] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics, 36(4), 2017. 6, 7

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012. 6

[21] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. arXiv preprint arXiv:2304.09987, 2023. 1, 3

[22] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14556–14565, 2021. 3

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. Advances in Neural Information Processing Systems, 33:15651–15663, 2020. 3, 6, 7

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

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1):99–106, 2021. 1, 2, 3, 4, 6, 7

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG), 41(4):1–15, 2022. 1, 3, 4, 6, 7, 8

[27] Arthur C Norman and PMA Moore. Implementing the new risch integration algorithm. In Proceedings of the 4th International Colloquium on Advanced Computing Methods in Theoretical Physics, pages 99–110, 1977. 2

[28] Robert H Risch. The problem of integration in finite terms. Transactions of the American Mathematical Society, 139:167–189, 1969. 2

[29] Robert H Risch. The solution of the problem of integration in finite terms. Bulletin of the American Mathematical Society, 76(3):605–608, 1970. 2

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 6

[31] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5459–5469, 2022. 1, 3, 7

[32] Ronald J Tallarida, Rodney B Murray, Ronald J Tallarida, and Rodney B Murray. Area under a curve: trapezoidal and simpson's rules. Manual of Pharmacologic Calculations: with Computer Programs, pages 77–81, 1987. 2

[33] Haithem Turki, Michael Zollhöfer, Christian Richardt, and Deva Ramanan. Pynerf: Pyramidal neural radiance fields. arXiv preprint arXiv:2312.00252, 2023. 1, 3, 6, 7

[34] 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, 7

[35] 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. 8

[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612, 2004. 6

[37] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16200–16209, 2022. 2, 3, 7

[38] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In Computer Graphics Forum, volume 41, pages 641–676. Wiley Online Library, 2022. 1

[39] 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. 1, 3

[40] 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. 8

[41] Lingyun Ye. Numerical quadrature: Theory and computation. PhD thesis, Dalhousie University, 2006. 2

[42] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5752–5761, 2021. 3

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018. 6