Exact-NeRF: An Exploration of a Precise Volumetric Parameterization for Neural Radiance Fields

Brian K. S. Isaac-Medina[1], Chris G. Willcocks[1], Toby P. Breckon[1, 2]
Department of {[1]Computer Science, [2]Engineering}, Durham University, UK
{brian.k.isaac-medina, christopher.g.willcocks, toby.breckon}@durham.ac.uk

Abstract

Neural Radiance Fields (NeRF) have attracted significant attention due to their ability to synthesize novel scene views with great accuracy. However, inherent to their underlying formulation, the sampling of points along a ray with zero width may result in ambiguous representations that lead to further rendering artifacts such as aliasing in the final scene. To address this issue, the recent variant mip-NeRF proposes an Integrated Positional Encoding (IPE) based on a conical view frustum. Although this is expressed with an integral formulation, mip-NeRF instead approximates this integral as the expected value of a multivariate Gaussian distribution. This approximation is reliable for short frustums but degrades with highly elongated regions, which arises when dealing with distant scene objects under a larger depth of field. In this paper, we explore the use of an exact approach for calculating the IPE by using a pyramid-based integral formulation instead of an approximated conical-based one. We denote this formulation as Exact-NeRF and contribute the first approach to offer a precise analytical solution to the IPE within the NeRF domain. Our exploratory work illustrates that such an exact formulation (Exact-NeRF) matches the accuracy of mip-NeRF and furthermore provides a natural extension to more challenging scenarios without further modification, such as in the case of unbounded scenes. Our contribution aims to both address the hitherto unexplored issues of frustum approximation in earlier NeRF work and additionally provide insight into the potential future consideration of analytical solutions in future NeRF extensions.

1. Introduction

Novel view synthesis is a classical and long-standing task in computer vision that has been thoroughly re-investigated via recent work on Neural Radiance Fields (NeRF) [20]. NeRF learns an implicit representation of a 3D scene from a set of 2D images via a Multi-Layer Perceptron (MLP) that predicts the visual properties of 3D points uniformly sampled along the viewing ray given its coordinates and viewing direction. This parameterization gives NeRF the dual ability to both represent 3D scenes and synthesize unseen views. In its original formulation, NeRF illustrates strong reconstruction performance for synthetic datasets comprising object-centric scenes and no background (bounded) and forward-facing real-world scenes. Among its applications, NeRF has been used for urban scene representation [25, 27, 29], human body reconstruction [3, 16], image processing [12, 17, 19] and physics [9, 14].

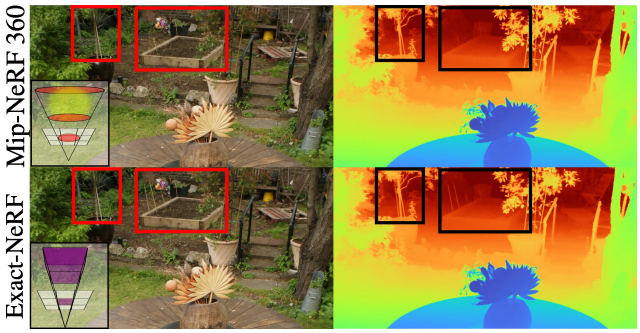

Figure 1. Comparison of Exact-NeRF (ours) with mip-NeRF 360 [2]. Our method is able to both match the performance and obtain superior depth estimation over a larger depth of field.

Nonetheless, the underlying sparse representation of 3D points learnt by the MLP may cause ambiguities that can lead to aliasing and blurring. To overcome these issues, Barron *et al*. proposed mip-NeRF [1], an architecture that uses cone tracing instead of rays. This architecture encodes conical frustums as the inputs of the MLP by approximating the integral of a sine/cosine function over a region in the space with a multivariate Gaussian. This reparameterization notably increases the reconstruction quality of multi-scale datasets. However, this approximation is only really valid for bounded scenes, where the conic frustums do not suffer from large elongations attributable to a large depth of field within the scene.

The NeRF concept has been extended to represent increasingly difficult scenes. For instance, mip-NeRF 360 [2] learns a representation of unbounded scenes with a central object by giving more capacity to points that are near the camera, modifying the network architecture and introducing a regularizer that penalizes 'floaters' (unconnected depth regions in free space) and other small unconnected regions. In order to model distant regions, mip-NeRF 360 transforms the multivariate Gaussians with a contraction function. This modification allows a better representation and outperforms standard mip-NeRF for an unbounded scenes dataset. However, the modification of the Gaussians requires attentive analysis to encode the correct information in the contracted space, which includes the linearization of the contraction function to accommodate the Gaussian approximations. This leads to a degraded performance of mip-NeRF 360 when the camera is far from the object. Additionally, mip-NeRF 360 struggles to render thin structures such as tree branches or bicycle rays.

Motivated by this, we present Exact-NeRF as an exploration of an alternative exact parameterization of underlying volumetric regions that are used in the context of mip-NeRF (Fig. 1). We propose a closed-form volumetric positional encoding formulation (Sec. 3) based on pyramidal frustums instead of the multivariate Gaussian approximation used by mip-NeRF and mip-NeRF 360. Exact-NeRF matches the performance of mip-NeRF on a synthetic dataset, but gets a sharper reconstruction around edges. Our approach can be applied without further modification to the contracted space of mip-NeRF 360. Our naive implementation of Exact-NeRF for the unbounded scenes of mip-NeRF 360 has a small decrease in performance, but it is able to get cleaner reconstructions of the background. Additionally, the depth map estimations obtained by Exact-NeRF are less noisy than mip-NeRF 360. Our key contribution is the formulation of a general integrated positional encoding framework that can be applied to any shape that can be broken into triangles (*i.e.*, a polyhedron). We intend that our work serves as a motivation to investigate different shapes and analytical solutions of volumetric positional encoding. The code is available at https://github.com/KostadinovShalon/exact-nerf.

2. Related Work

Already numerous work has focused on improving NeRF since its original inception [20], such as decreasing the training time [5, 6, 8, 11, 30], increasing the synthesis speed [10,26,31], reducing the number of input images [23] and improving the rendering quality [1, 2, 15, 18, 28, 32]. With the latter, one of the focuses is to change the positional encoding to account for the volumetric nature of the regions that contribute to pixel rendering [1,2].

2.1. Positional Encoding

NeRF uses a positional encoding (PE) on the raw coordinates of the input points in order to induce the network to learn higher-frequency features [24]. However, the sampled points in NeRF are intended to represent a region in the volumetric space. This can lead to ambiguities that may cause aliasing. In this sense, mip-NeRF [1] uses a volumetric rendering by casting cones instead of rays, changing the input of the MLP from points to cone frustums. These regions are encoded using an integrated positional encoding (IPE), which aims to integrate the PE over the cone frustums. Given that the associated integral has no closed-form solution, they formulate the IPE as the expected value of the positional encoding in a 3D Gaussian distribution centred in the frustum. The IPE reduces aliasing by reducing the ambiguity of single-point encoding. Mip-NeRF 360 [2] uses a contracted space representation to extend the mip-NeRF parameterization to 360° unbounded scenes, since they found that the approximation given in mip-NeRF degrades for elongated frustums which arise in the background. Additionally, and similar to DONeRF [22], mip-NeRF 360 samples the intervals of the volumetric regions using the inverse of the distance in order to assign a bigger capacity to nearer objects. By contrast, in this work we explore the use of pyramid-based frustums in order to enable an exact integral formulation of the IPE which can be applied for both bounded and unbounded scenes alike.

2.2. NeRF and Mip-NeRF parameterization

NeRF uses an MLP f with parameters Θ to get the colour $\mathbf{c} \in \mathbb{R}^3$ and density $\sigma \in [0, +\infty)$ given a point $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\hat{\mathbf{v}} \in S^2$, where S^2 is the unit sphere, such that:

$$(\mathbf{c}, \sigma) = f(\mathbf{x}, \hat{\mathbf{v}}; \Theta)\,, \tag{1}$$

whereby NeRF uses the positional encoding

$$\begin{aligned}\gamma(\mathbf{x}) = \big[&\sin(2^0\mathbf{x}), \dots, \sin(2^{L-1}\mathbf{x}),\\ &\cos(2^0\mathbf{x}), \dots, \cos(2^{L-1}\mathbf{x})\big]^\top,\end{aligned} \tag{2}$$

and $\gamma : \mathbb{R} \to \mathbb{R}^{2L}$ is applied to each coordinate of $\mathbf{x}$ and each component of $\hat{\mathbf{v}}$ independently. The sampling strategy of NeRF consists of sampling random points along the ray that passes through a pixel. This ray is represented by $\mathbf{r}(t) = t\mathbf{d} + \mathbf{o}$, where $\mathbf{o}$ is the camera position and $\mathbf{d}$ is the vector that goes from the camera centre to the pixel in the image plane. The ray is divided into N intervals and the points $\mathbf{r}(t_i)$ are drawn from a uniform distribution over each interval, such that:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right], \tag{3}$$

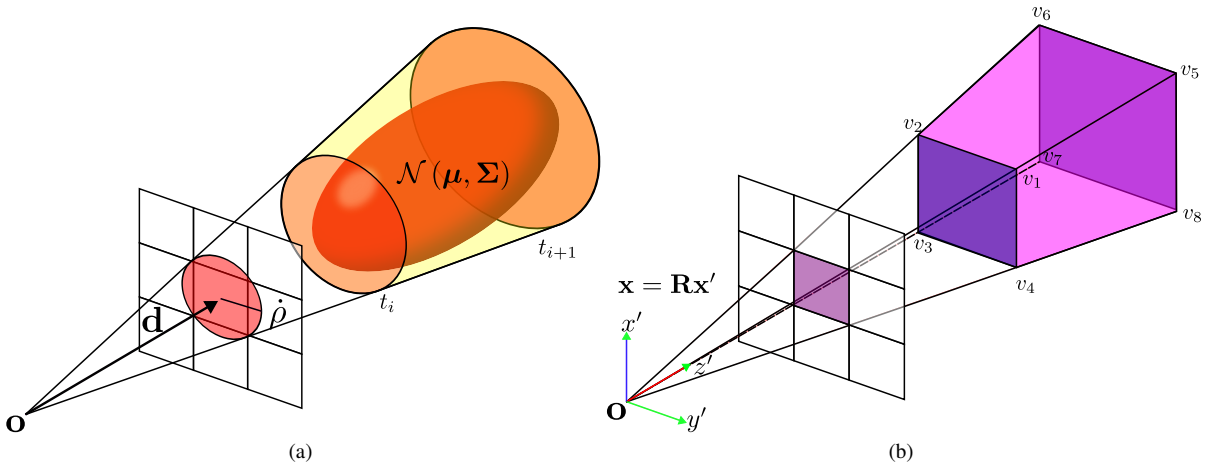

Figure 2. Cone and pyramid tracing for volumetric NeRF parameterizations. (a) Mip-NeRF [1] uses cone frustums to parameterize a 3D region. Since the IPE of these frustums does not have a closed-form solution, it is approximated by modelling the frustum as a multivariate Gaussian. (b) Exact-NeRF casts a square pyramid instead of a cone, allowing for an exact parameterization of the IPE by using the vertices v_i of the frustum and the pose parameters $\mathbf{o}$ and $\mathbf{R}$.

where t_n and t_f are the near and far planes. In this sense, the colour and density of each point over the ray are obtained by $(\mathbf{c}_i, \sigma_i) = f(\gamma(\mathbf{r}(t_i)), \gamma(\mathbf{d}/\|\mathbf{d}\|); \Theta)$.

Finally, the pixel colour $\hat{C}(\mathbf{r})$ is obtained using numerical quadrature,

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \\ T_i &= \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right) , \end{aligned} \tag{4}$$

where $\delta_i = t_{i+1} - t_i$. This process is carried out hierarchically by using coarse $\hat{C}_c$ and fine $\hat{C}_f$ samplings, where the 3D points in the latter are drawn from the PDF formed by the weights of the density values of the coarse sampling. The loss is then the combination of the mean-squared error of the coarse and fine renderings for all rays $\mathbf{r} \in \mathcal{R}$, *i.e.*,

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \|\hat{C}_c(\mathbf{r}) - C(\mathbf{r})\|_2^2 + \|\hat{C}_f(\mathbf{r}) - C(\mathbf{r})\|_2^2 . \tag{5}$$

Here we find mip-NeRF [1] is similar to NeRF, but it utilises cone tracing instead of ray tracing. This change has the direct consequence of replacing ray intervals by conical frustums $F(\mathbf{d}, \mathbf{o}, \dot{\rho}, t_i, t_{i+1})$, where $\dot{\rho}$ is the radius of the circular section of the cone at the image plane (Fig. 2a). This leads to the need for a new positional encoding that summarizes the function in Eq. (2) over the region defined by the frustum. The proposed IPE is thus given by:

$$\gamma_I(\mathbf{d}, \mathbf{o}, \dot{\rho}, t_i, t_{i+1}) = \frac{\iiint_F \gamma(\mathbf{x}) dV}{\iiint_F dV} . \tag{6}$$

Since the integral in the numerator of Eq. (6) has no closed-form solution, mip-NeRF proposes to approximate it by considering the cone frustums as multivariate Gaussians. Subsequently, the approximated IPE γ^* is given by:

$$\begin{aligned} \gamma^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \mathbb{E}_{x \sim \mathcal{N}(\mathbf{P}\boldsymbol{\mu}, \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\top)} [\gamma(\mathbf{x})] \\ &= \begin{bmatrix} \sin(\mathbf{P}\boldsymbol{\mu}) \circ \exp(-(1/2)\mathrm{diag}(\mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\top)) \\ \cos(\mathbf{P}\boldsymbol{\mu}) \circ \exp(-(1/2)\mathrm{diag}(\mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\top)) \end{bmatrix} , \end{aligned} \tag{7}$$

where $\boldsymbol{\mu} = \mathbf{o} + \mu_t \mathbf{d}$ is the centre of the Gaussian for a frustum defined by $\mathbf{o}$ and $\mathbf{d}$ with mean distance along the ray μ_t, $\boldsymbol{\Sigma}$ is the covariance matrix, $\circ$ denotes element-wise product and:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 & & 2^{L-1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & \dots & 0 & 2^{L-1} & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & & 0 & 0 & 2^{L-1} \end{bmatrix}^\top . \tag{8}$$

This formulation was empirically shown to be accurate for bounded scenes where a central object is the main part of the scene and no background information is present. However, the approximation deteriorates for highly elongated frustums. To avoid this, mip-NeRF 360 [2] instead uses a contracted space where points that are beyond the unit sphere are mapped using the function:

$$f(\mathbf{x}) = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \leq 1 \\ \left(2 - \frac{1}{\|\mathbf{x}\|}\right) \frac{\mathbf{x}}{\|\mathbf{x}\|} & \|\mathbf{x}\| > 1 \end{cases} . \tag{9}$$

Subsequently, the new $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ values are given by $f(\boldsymbol{\mu})$ and $\mathbf{J}_f(\boldsymbol{\mu})\boldsymbol{\Sigma}\mathbf{J}_f(\boldsymbol{\mu})^\top$, where $\mathbf{J}_f$ is the Jacobian matrix of f. Empirically, this re-parameterization now allows learning the representation of scenes with distant backgrounds (*i.e.* over a longer depth of field).

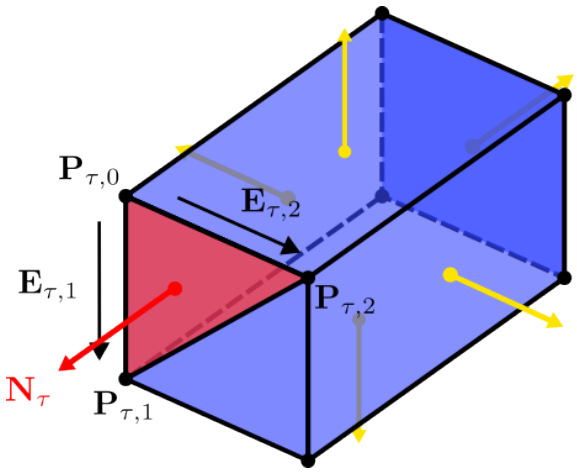

Figure 3. Parameterization of triangular faces. The vertices are sorted counter-clockwise, so the normal vector to their plane points outside the frustum.

3. Exact-NeRF

In this paper, we present Exact-NeRF as an exploration of how the IPE approximations of earlier work [1, 2] based on conic parameterization can be replaced with a square pyramid-based formulation in order to obtain an exact IPE γ_E, as shown in Fig. 2. The motivation behind this formulation is to match the volumetric rendering with the pixel footprint, which in turn is a rectangle.

3.1. Volume of pyramidal frustums

A pyramidal frustum can be defined by a set of 8 vertices $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{8}$ and 6 quadrilateral faces $\mathcal{F} = \{f_j\}_{j=1}^{6}$. In order to get the volume in the denominator of Eq. (6), we use the divergence theorem:

$$\iiint \nabla \cdot F dV = \oiint_{\partial S} F \cdot d\mathbf{S}\,, \tag{10}$$

with $F = \frac{1}{3}[x, y, z]^\top$, yielding to the solution for the volume as:

$$V = \iiint dV = \frac{1}{3} \oiint_{\partial S} [x, y, z]\, d\mathbf{S}\,. \tag{11}$$

Without losing generality, we divide each face into triangles, giving a set of triangular faces $\mathcal{T}$ such that the polyhedra formed by faces $\mathcal{F}$ and $\mathcal{T}$ are the same. Each triangle τ is defined by three points $\mathbf{P}_{\tau,0}, \mathbf{P}_{\tau,1}$ and $\mathbf{P}_{\tau,2}$, with $\mathbf{P}_{\tau,i} \in \mathcal{V}$, such that the cross product of the edges $\mathbf{E}_{\tau,1} = \mathbf{P}_{\tau,1} - \mathbf{P}_{\tau,0}$ and $\mathbf{E}_{\tau,2} = \mathbf{P}_{\tau,2} - \mathbf{P}_{\tau,0}$ points outside the frustum (Fig. 3). As a result, Eq. (11) equates to the sum of the surface integral for each triangle $\tau \in \mathcal{T}$,

$$V = \frac{1}{3} \sum_{\tau \in \mathcal{T}} \iint_{\tau} [x, y, z]\, d\mathbf{S}\,. \tag{12}$$

The points lying in the triangle $\triangle \mathbf{P}_{\tau,0}\mathbf{P}_{\tau,1}\mathbf{P}_{\tau,2}$ can hence be parameterized as:

$$\mathbf{P}_\tau(u, v) = \mathbf{P}_{\tau,0} + u\mathbf{E}_{\tau,1} + v\mathbf{E}_{\tau,2}\,, \tag{13}$$

such that $0 \le u \le 1, 0 \le v \le 1$ and $u + v \le 1$. The differential term of Eq. (12) is then:

$$d\mathbf{S} = \left(\frac{\partial \mathbf{P}_\tau}{\partial u} \times \frac{\partial \mathbf{P}_\tau}{\partial v}\right) dudv \tag{14}$$

$$d\mathbf{S} = (\mathbf{E}_{\tau,1} \times \mathbf{E}_{\tau,2})\, dudv \triangleq \mathbf{N}_\tau dudv\,. \tag{15}$$

By substituting Eq. (15) into Eq. (12), and noting that $[x, y, z] = \mathbf{P}_\tau(u, v)$, we obtain:

$$V = \frac{1}{3} \sum_{\tau \in \mathcal{T}} \int_0^1 \int_0^{1-v} \mathbf{P}_\tau(u, v)^\top \mathbf{N}_\tau dudv\,. \tag{16}$$

Since the dot product of any point $\mathbf{P}_\tau$ in a face τ with a vector $\mathbf{N}_\tau$ normal to τ is constant, the product inside the integral of Eq. (16) is constant. Subsequently, $\mathbf{P}_\tau(u, v)$ can be replaced with any point, such as $\mathbf{P}_{\tau,0}$. Finally, the required volume is obtained as:

$$\begin{aligned} V &= \frac{1}{3} \sum_{\tau \in \mathcal{T}} \mathbf{P}_{\tau,0}^\top \mathbf{N}_\tau \int_0^1 \int_0^{1-v} dudv \\ &= \frac{1}{6} \sum_{\tau \in \mathcal{T}} \mathbf{P}_{\tau,0}^\top \mathbf{N}_\tau\,. \end{aligned} \tag{17}$$

3.2. Integration over the PE Function

Following from earlier, we can obtain the numerator of the IPE in Eq. (6) using the divergence theorem. We will base our analysis on the sine function and the x coordinate, *i.e.*, $\gamma(x) = \sin(2^l x)$. Substituting $F = \left[-\frac{1}{2^l}\cos(2^l x), 0, 0\right]^\top$ in Eq. (10) we obtain:

$$\iiint \sin(2^l x) dV = \oiint_{\partial S} \left[-\frac{1}{2^l}\cos(2^l x), 0, 0\right] d\mathbf{S}\,. \tag{18}$$

Following the same strategy of dividing the surface into triangular faces as in the earlier volume calculation, Eq. (18) can be written as:

$$\iiint \sin(2^l x) dV = \sum_{\tau \in \mathcal{T}} \frac{1}{2^l} \sigma_{x,\tau} \mathbf{N}_\tau \cdot \hat{\mathbf{i}}\,, \tag{19}$$

where $\hat{\mathbf{i}}$ is the unit vector in the x direction and:

$$\sigma_{x,\tau} = \int_0^1 \int_0^{1-v} -\cos(2^l x_\tau(u, v)) dudv\,. \tag{20}$$

From Eq. (13), the x coordinate can be parameterized as:

$$x_\tau(u, v) = x_{\tau,0} + u(x_{\tau,1} - x_{\tau,0}) + v(x_{\tau,2} - x_{\tau,0})\,. \tag{21}$$

Substituting Eq. (21) in Eq. (20) and solving the integral, we obtain:

$$\sigma_{x,\tau} = \frac{1}{2^{2l}} \left(\frac{\cos(2^l x_{\tau,0})}{(x_{\tau,0} - x_{\tau,1})(x_{\tau,0} - x_{\tau,2})} + \frac{\cos(2^l x_{\tau,1})}{(x_{\tau,1} - x_{\tau,0})(x_{\tau,1} - x_{\tau,2})} + \frac{\cos(2^l x_{\tau,2})}{(x_{\tau,2} - x_{\tau,0})(x_{\tau,2} - x_{\tau,1})} \right). \tag{22}$$

Furthermore, Eq. (22) can be written as:

$$\sigma_{x,\tau} = \frac{1}{2^{2l}} \frac{\det\left(\begin{bmatrix}\mathbf{1} & \boldsymbol{x}_\tau & \cos(2^l \boldsymbol{x}_\tau)\end{bmatrix}\right)}{\det\left(\begin{bmatrix}\mathbf{1} & \boldsymbol{x}_\tau & \boldsymbol{x}_\tau^{\circ 2}\end{bmatrix}\right)}, \tag{23}$$

where $\mathbf{1} = [1, 1, 1]^\top$, $\boldsymbol{x}_\tau = [x_{\tau,0}, x_{\tau,1}, x_{\tau,2}]^\top$ and $(\cdot)^{\circ n}$ is the element-wise power.

In general, we can also obtain the expression in Eq. (19) for the k-th coordinate of $\mathbf{x}$ as:

$$\iiint \sin(2^l \mathbf{x}_k) dV = \frac{1}{2^{3l}} \sum_{\tau \in \mathcal{T}} \sigma_{k,\tau} \mathbf{N}_\tau \cdot \mathbf{e}_k, \tag{24}$$

$$\sigma_{k,\tau} = \frac{\det\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}_\tau^\top \mathbf{e}_k & \cos(2^l \mathbf{X}_\tau^\top \mathbf{e}_k)\end{bmatrix}\right)}{\det\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}_\tau^\top \mathbf{e}_k & (\mathbf{X}_\tau^\top \mathbf{e}_k)^{\circ 2}\end{bmatrix}\right)}, \tag{25}$$

where $\mathbf{X}_\tau = \begin{bmatrix}\mathbf{P}_{\tau,0} & \mathbf{P}_{\tau,1} & \mathbf{P}_{\tau,2}\end{bmatrix}$ and $\mathbf{e}_k$ are the vectors that form the canonical basis in $\mathbb{R}^3$. Similarly, the integral over the cosine function is defined as:

$$\iiint \cos(2^l \mathbf{x}_k) dV = \frac{1}{2^{3l}} \sum_{\tau \in \mathcal{T}} \xi_{k,\tau} \mathbf{N}_\tau \cdot \mathbf{e}_k, \tag{26}$$

where:

$$\xi_{k,\tau} = -\frac{\det\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}_\tau^\top \mathbf{e}_k & \sin(2^l \mathbf{X}_\tau^\top \mathbf{e}_k)\end{bmatrix}\right)}{\det\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}_\tau^\top \mathbf{e}_k & (\mathbf{X}_\tau^\top \mathbf{e}_k)^{\circ 2}\end{bmatrix}\right)}. \tag{27}$$

Finally, we get the exact IPE (EIPE) of the frustum used by our Exact-NeRF approach by dividing Eqs. (24) and (26) by Eq. (17) as follows:

$$\gamma_E(\mathbf{x}, l; \mathcal{V}) = \frac{6}{2^{3l}} \begin{bmatrix} \frac{\sum_{\tau \in \mathcal{T}} \boldsymbol{\sigma}_\tau \circ \mathbf{N}_\tau}{\sum_{\tau \in \mathcal{T}} \mathbf{P}_{\tau,0}^\top \mathbf{N}_\tau} \\ \frac{\sum_{\tau \in \mathcal{T}} \boldsymbol{\xi}_\tau \circ \mathbf{N}_\tau}{\sum_{\tau \in \mathcal{T}} \mathbf{P}_{\tau,0}^\top \mathbf{N}_\tau} \end{bmatrix}, \tag{28}$$

where $\boldsymbol{\sigma}_\tau = \begin{bmatrix}\sigma_{1,\tau} & \sigma_{2,\tau} & \sigma_{3,\tau}\end{bmatrix}^\top$ and $\boldsymbol{\xi}_\tau = \begin{bmatrix}\xi_{1,\tau} & \xi_{2,\tau} & \xi_{3,\tau}\end{bmatrix}^\top$. It's worth mentioning that Eq. (28) fails when a coordinate value repeats in any of the points of a triangle (*i.e.*, there is a triangle τ such that $\mathbf{P}_{\tau,i} = \mathbf{P}_{\tau,j}$ for a $i \neq j$). For these cases, *l'Hopital's rule* can be used to evaluate this limit (see Supplementary Material).

Despite starting our analysis with squared pyramids, it can be noted that Eq. (28) is true for any set of vertices $\mathcal{V}$, meaning that this parameterization can be applied for any shape with known vertices. This is particularly useful for scenarios where the space may be deformed and frustums may not be perfect pyramids, such as in mip-Nerf 360 [2]. Additionally, it can be noted that our EIPE is multiplied by a factor of 2^{-3l}, meaning that when $L \to \infty$ then $\gamma_E \to 0$ which hence makes our implementation robust to large values of L. This property of our Exact-NeRF formulation is consistent with that of the original mip-NeRF [1].

4. Implementation Details

Exact-NeRF is implemented using the original code of mip-NeRF, which is based on JAXNeRF [4]. Apart from the change of the positional encoding, no further modification is made. We use the same sampling strategy of ray intervals defined in Eq. (3), but sampling $N + 1$ points to define N intervals. In order to obtain the vertices of the pyramid frustums, we use the coordinates of the corners of each pixel and multiply them by the t_i values to get the front and back faces of the frustums. Double precision (64-bit float) is used for calculating the EIPE itself, as it relies upon arithmetic over very low numerical decimals that are otherwise prone to numerical precision error (see Eq. (22)). After calculation, the EIPE result is transformed back to single precision (32-bit float).

We compare our implementation of Exact-NeRF against the original mip-NeRF baseline on the benchmark Blender dataset [20], down-sampled by a factor of 2. We follow a similar training strategy as in mip-NeRF: training both models for 800k iterations (instead of 1 million, as we observed convergence at this point) with a batch size of 4096 using Adam optimization [13] with a logarithmically annealed learning rate, $5 \times 10^{-4} \to 5 \times 10^{-6}$. All training is carried out using 2 $\times$ NVIDIA Tesla V100 GPU per scene.

Additionally, we compare the use of the EIPE against mip-NeRF 360 on the dataset of Barron *et al.* [2]. Similarly, we used the reference code from MultiNeRF [21], which contains an implementation of mip-NeRF 360 [2], RefNeRF [28] and RawNeRF [19]. Pyramidal frustum vertices are contracted using Eq. (9) and the EIPE is obtained using the Eq. (28) with the mapped vertices. We trained using a batch size of 8192 for 500k iterations using 4 $\times$ NVIDIA Tesla V100 GPU per scene. Aside from the use of the EIPE, all other settings remained unchanged from mip-NeRF 360 [2].

5. Results

Mean PSNR, SSIM and LPIPS [33] metrics are reported for our Exact-NeRF approach, mip-NeRF [1] and mip-NeRF 360 [2]. Additionally, we also report the DISTS [7] metric since it provides another perceptual quality measurement. Similar to mip-NeRF, we also report an average metric: the

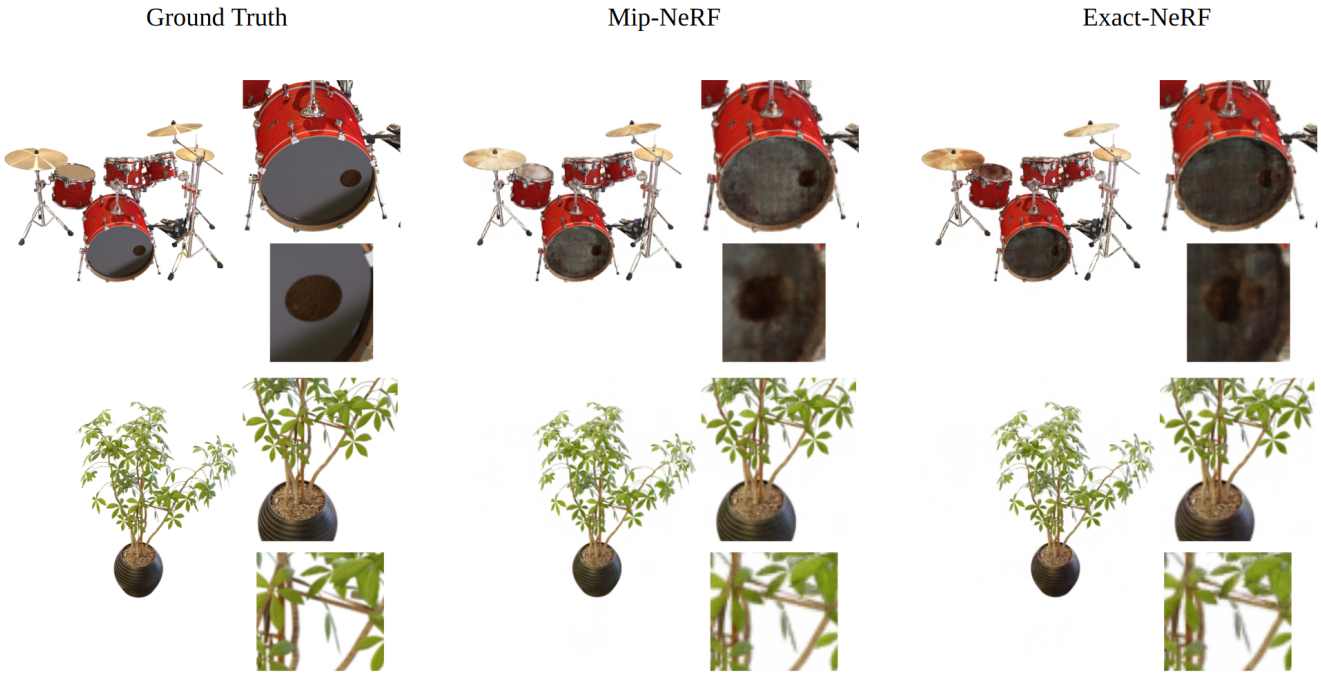

Figure 4. Qualitative comparison between mip-NeRF and Exact-NeRF (ours) for the blender dataset. Our method matches the mip-NeRF rendering capability but also produces slightly sharper renderings (see the bass drum hole and the back leaves of the ficus).

Model	PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	Avg ↓
Mip-NeRF	**34.766**	**0.9706**	0.0675	0.0878	**0.0242**
Exact-NeRF (ours)	34.707	0.9705	**0.0667**	**0.0822**	**0.0242**

Table 1. Quantitative results comparing mip-NeRF and Exact-NeRF performance on the Blender dataset.

Model	PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	Avg ↓
Mip-NeRF 360	**27.325**	**0.7942**	**0.6559**	**0.2438**	**0.1077**
Exact-NeRF (ours)	27.230	0.7881	0.6569	0.2452	0.1088

Table 2. Comparison of the performance of Exact-NeRF with mip-NeRF 360 on the unbounded dataset of Barron *et al*. [2].

geometric mean of the MSE = $10^{-\text{PSNR}/10}$, $\sqrt{1-\text{SSIM}}$, the LPIPS and the DISTS.

Blender dataset: In Tab. 1 we present a quantitative comparison between Exact-NeRF and mip-NeRF. It can be observed that our method matches the reconstruction performance of mip-NeRF, with a marginal decrease of the PSNR and SSIM and an increment in the LPIPS and DISTS metrics, but with identical average performance. This small decrement in the PSNR and SSIM metrics can be explained by the loss of precision in the calculation of small quantities involved in the EIPE. Alternative formulations using the same idea could be used (see Supplementary Material), but the intention of Exact-NeRF is to create a general approach for any volumetric positional encoding using the vertices of the volumetric region. Fig. 4 shows a qualitative comparison between mip-NeRF and Exact-NeRF. It can be observed that Exact-NeRF is able to match the reconstruction performance of mip-NeRF. A closer examination reveals that Exact-NeRF creates sharper reconstructions in some regions, such as the hole in the bass drum or the leaves in the ficus, which is explained by mip-NeRF approximating the conical frustums as Gaussians. This is consistent with the increase in the LPIPS and DISTS, which are the perceptual similarity metrics.

Mip-NeRF 360 dataset: Tab. 2 shows the results for the unbounded mip-NeRF 360 dataset. Despite Exact-NeRF having marginally weaker reconstruction metrics, it shows a competitive performance without any changes to the implementation of the EIPE used earlier with the bounded blender dataset, *i.e*., the contracted vertices were directly used without any further simplification or linearization, as in mip-NeRF 360 [2]. Similar to the blender dataset results, this decrement can be explained with the loss of precision, which suggests that an alternative implementation of Eq. (28) may be needed. A qualitative comparison is shown in Fig. 5. It can be observed that tiny vessels are more problematic for Exact-NeRF (Fig. 5a), which can be explained again by the loss of precision. However, it is noted in Fig. 5b that the reconstruction of far regions in mip-NeRF 360 is noisier than Exact-NeRF (see Fig. 5b, grill and the car), which is a consequence of the poor approximation of the Gaussian region for far depth of field objects in the

Figure 5. Qualitative comparison between mip-NeRF 360 and Exact-NeRF (ours). (a) Our model, similar to mip-NeRF, struggles with tiny vessels. (b) Exact-NeRF shows cleaner renderings and (c) higher quality background reconstruction.

scene. Fig. 5c reveals another example of a clearer region in the Exact-NeRF reconstruction for the background detail. Fig. 6 shows snapshots of the depth estimation for the bicycle, bonsai and garden scenes. Consistent with the colour reconstructions, some background regions have a more detailed estimation. It is also noticed (not shown) that despite Exact-NeRF having a smoother depth estimation, it may show some artifacts in the form of straight lines, which may be caused by the shape of the pyramidal frustums. It is worth reminding that our implementation of the EIPE in mip-NeRF 360 is identical to the EIPE in mip-NeRF.

Impact of Numerical Underflow As seen in Sec. 3, Exact-NeRF may suffer from numerical underflow when the difference of a component of two points $\Delta = x_{\tau,i} - x_{\tau,j}$ is too close to zero ($\Delta \to 0$). In the case of this difference being precisely zero, the limit can be found using *l'Hopital's rule*, as is further developed in Appendix A.1. However, if this value is not zero but approximately zero, numerical underflow could lead to exploding values in Eq. (22). This error hinders the training of the MLP since the IPE is bounded to the interval $[-1, 1]$ by definition (Eq. (6)). An example of the effect of numerical underflow in our method applied under the mip-NeRF 360 framework is shown in Fig. 7. The black lines are the location of such instances where underflow occurs. The curvature of these lines is a direct consequence of the contracted space used in mip-NeRF 360. In order to eliminate this effect, we use double precision for the calculation of the EIPE. Additionally, all differences of a coordinate which are less than 1×10^{-6} are set to zero and reformulated using *l'Hopital's rule*.

6. Conclusion

In this work, we present Exact-NeRF, a novel precise volumetric parameterization for neural radiance fields (NeRF). In contrast to conical frustum approximation via a multivariate Gaussian in mip-NeRF [1], Exact-NeRF uses a novel pyramidal parameterization to encode 3D regions using an Exact Integrated Positional Encoding (EIPE). The EIPE applies the divergence theorem to compute the exact value of the positional encoding (an array of sine and cosines) in a pyramidal frustum using the coordinates of the vertices that define the region. Our proposed EIPE methodology can be applied to any such architecture that performs volumetric positional encoding from simple knowledge of

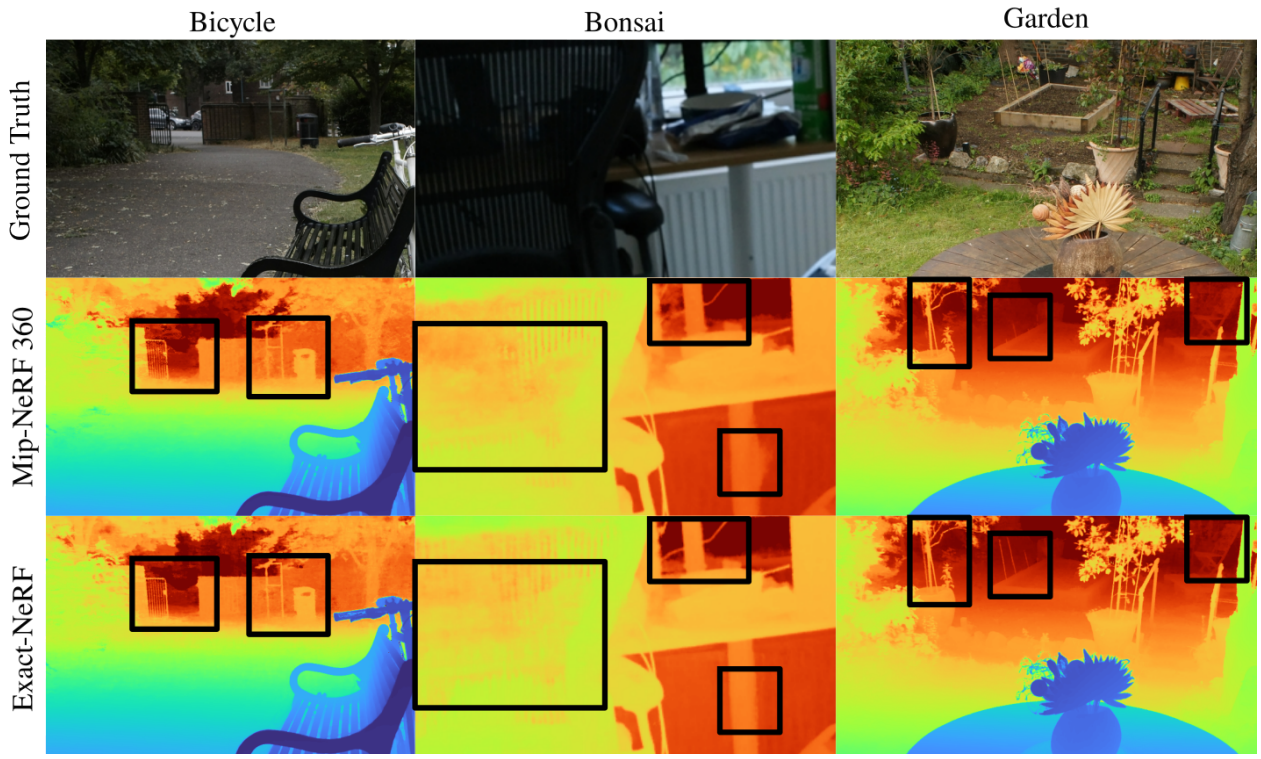

Figure 6. Depth estimation for mip-NerF 360 and Exact-NeRF. Our approach shows better depth estimations for background regions (highlighted in the black boxes), although some artifacts in form of straight lines may appear, which is inherent in our pyramidal shapes.

Figure 7. Numerical underflow artifacts in Exact-NeRF.

the pyramidal frustum vertices without the need for further processing.

We compare Exact-NeRF against mip-NeRF on the blender dataset, showing a matching performance with a marginal decrease in PSNR and SSIM but an overall improvement in the perceptual metric, LPIPS. Qualitatively our approach exhibits slightly cleaner and sharper reconstructions of edges than mip-NeRF [1].

We similarly compare Exact-NeRF with mip-NeRF 360 [2]. Despite Exact-NeRF showing a marginal decrease in performance metrics, it illustrates the capability of the EIPE on a different architecture without further modification. Exact-NeRF obtains sharper renderings of distant (far depth of field) regions and areas where mip-NeRF 360 presents some noise, but it fails to reconstruct tiny vessels in near regions. The qualitative depth estimations maps also confirm these results. The marginal decrease in performance of our Exact-NeRF method can be attributed to numerical underflow and some artifacts caused by the choice of a step-function-based square pyramidal parameterization. In addition, our results suggest using a combined encoding such that the EIPE is used for distance objects, where it is more stable and accurate. Although alternative solutions can be obtained by restricting the analysis to rectangular pyramids, our aim is to introduce a general framework that can be applied to any representation of a 3D region with known vertices. The investigation of more stable representations and the performance of different shapes for modelling 3D regions under a neural rendering context remains an area for future work.

Acknowledgments

This work is partially supported by the Mexican Council of Science and Technology (CONACyT).

References

[1] 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 Conf. Comput. Vis. Pattern Recog.*, pages 5855–5864, 2021. 1, 2, 3, 4, 5, 7, 8

[2] 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 Conf. Comput. Vis. Pattern Recog.*, pages 5470–5479, 2022. 1, 2, 3, 4, 5, 6, 8

[3] Abril Corona-Figueroa, Jonathan Frawley, Sam Bond Taylor, Sarath Bethapudi, Hubert P. H. Shum, and Chris G. Willcocks. Mednerf: Medical neural radiance fields for reconstructing 3d-aware ct-projections from a single x-ray. In *Proceedings of the 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society*, pages 3843–3848, 2022. 1

[4] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF. https://github.com/google-research/google-research/tree/master/jaxnerf, 2020. 5

[5] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12882–12891, June 2022. 2

[6] Arnab Dey, Yassine Ahmine, and Andrew I. Comport. Mip-NeRF RGB-d: Depth assisted fast neural radiance fields. *Journal of WSCG*, 30(1-2):34–43, 2022. 2

[7] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *Proceedingf the IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 5

[8] 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 Conf. Comput. Vis. Pattern Recog.*, pages 5501–5510, June 2022. 2

[9] Shanyan Guan, Huayu Deng, Yunbo Wang, and Xiaokang Yang. Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *Proceedings of the Int. Conf. on Mach. Learning*, 2022. 1

[10] 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 Int. Conf. Comput. Vis.*, pages 5875–5884, October 2021. 2

[11] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12902–12911, June 2022. 2

[12] Xin Huang, Qi Zhang, Ying Feng, Hongdong Li, Xuan Wang, and Qing Wang. Hdr-nerf: High dynamic range neural radiance fields. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 18398–18408, 2022. 1

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the Int. Conf. Learn. Represent.*, 2015. 5

[14] Aviad Levis, Pratul P Srinivasan, Andrew A Chael, Ren Ng, and Katherine L Bouman. Gravitationally lensed black hole emission tomography. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19841–19850, 2022. 1

[15] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the Int. Conf. Comput. Vis.*, 2021. 2

[16] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *Proceedings of the ACM Trans. Graph.*, 34(6):248:1–248:16, Oct. 2015. 1

[17] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12861–12870, 2022. 1

[18] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7210–7219, 2021. 2

[19] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16190–16199, 2022. 1, 5

[20] 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, 5

[21] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF. https://github.com/google-research/multinerf, 2022. 5

[22] 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*, 40(4), 2021. 2

[23] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2

[24] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the Int. Conf. on Mach. Learning*, pages 5301–5310. PMLR, 2019. 2

[25] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In

Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog., pages 12932–12942, 2022. 1

[26] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Adv. Neural Inform. Process. Syst.*, 34:19313–19325, 2021. 2

[27] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8248–8258, 2022. 1

[28] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2, 5

[29] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *Proceedings of the Eur. Conf. Comput. Vis.*, 2022. 1

[30] Guo-Wei Yang, Wen-Yang Zhou, Hao-Yang Peng, Dun Liang, Tai-Jiang Mu, and Shi-Min Hu. Recursive-nerf: An efficient and dynamically growing nerf. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–14, 2022. 2

[31] 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 Int. Conf. Comput. Vis.*, pages 5752–5761, October 2021. 2

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

[33] 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 Conf. Comput. Vis. Pattern Recog.*, 2018. 5