NeRFocus: Neural Radiance Field for 3D Synthetic Defocus

Yinhuai Wang[1], Shuzhou Yang[1], Yujie Hu[1], and Jian Zhang[1,2]

Peking University, Shenzhen Graduate School[1]
Peng Cheng Laboratory[2]

Abstract. Neural radiance fields (NeRF) bring a new wave for 3D interactive experiences. However, as an important part of the immersive experiences, the defocus effects have not been fully explored within NeRF. Some recent NeRF-based methods generate 3D defocus effects in a post-process fashion by utilizing multiplane technology. Still, they are either time-consuming or memory-consuming. This paper proposes a novel thin-lens-imaging-based NeRF framework that can directly render various 3D defocus effects, dubbed NeRFocus. Unlike the pinhole, the thin lens refracts rays of a scene point, so its imaging on the sensor plane is scattered as a circle of confusion (CoC). A direct solution sampling enough rays to approximate this process is computationally expensive. Instead, we propose to inverse the thin lens imaging to explicitly model the beam path for each point on the sensor plane and generalize this paradigm to the beam path of each pixel, then use the frustum-based volume rendering to render each pixel's beam path. We further design an efficient probabilistic training (p-training) strategy to simplify the training process vastly. Extensive experiments demonstrate that our NeRFocus can achieve various 3D defocus effects with adjustable camera pose, focus distance, and aperture size. Existing NeRF can be regarded as our special case by setting aperture size as zero to render large depth-of-field images. Despite such merits, NeRFocus does not sacrifice NeRF's original performance (e.g., training and inference time, parameter consumption, rendering quality), which implies its great potential for broader application and further improvement.

Keywords: Neural Radiance Field, Defocus effects

1 Introduction

A small camera aperture took an image usually has a very large depth-of-field (DOF), making all the objects in the scene clear and in focus. A shallow DOF is obtained if a large camera aperture is used to capture the same scene. In this condition, objects near the focal plane appear clear, while those far from the focal plane are optically blurred. This phenomenon is known as defocus effects and is prevalent in film-making and portrait photography.

Defocus effects are depth-dependent, and if we strictly follow the principle of lens imaging [6], [12], the computation will be expensive. Previous image-based methods for rendering defocus usually simplify the lens imaging with the

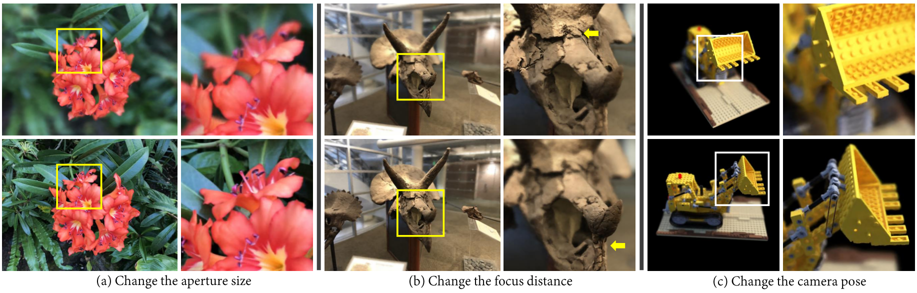

(a) Change the aperture size
(b) Change the focus distance
(c) Change the camera pose

Fig. 1. We present a novel framework that extends a neural radiance field (NeRF) to directly render various 3D defocus effects without post-process. (a) We can change the aperture size to raise or reduce defocus effects. (b) Or, change the focus distance to move the depth-of-field (DOF) forward or backward, as the yellow arrows denote. (c) Our framework fully inherits Nerf's novel view synthesis and can synthesize high-quality 3D defocus effects with disentangled control on aperture, focus distance, and camera pose. **See the supplemented video for demonstrations**.

depth-dependent blur operations on discretized depth layers. Different from these methods, NeRF [21] renders images by implicitly modeling the scene representation, thus naturally containing depth information for defocus effects. However, a fundamental barrier for NeRF to render defocus effects is that its imaging is based on a pinhole model, by which the whole scene is in sharp focus. To solve this limitation, we propose NeRFocus, a novel NeRF-based framework that approximates the imaging of a thin lens model.

In a thin lens model, a scene point scatters as a circle-of-confusion (CoC) on the sensor plane. The CoC diameter can be derived from the basic lens equation. We inverse the thin lens imaging to form the beam path for each point on the sensor plane as their beam path and generalize this paradigm to the beam path of each pixel. Specifically, we incorporate the merit of mip-NeRF [2] that extend a circle rather than a point as the receptive field of a pixel. We calculate the beam path for each point inside the circle following the inverse thin lens imaging. These beam paths finally overlap as a composite cone representing the beam path of a pixel.

To render the composite cone using volume rendering [11], [18], we divide it as frustums and assume the "importance" distribution inside each frustum to be a 3D Gaussian, by which we can calculate the expectation of radiance and density inside each frustum. To simplify the computation, we make a critical assumption that this expectation can be directly predicted by a multilayer perceptron (MLP) given the expected value of positional encoding (PE) as input. Hence, the critical problem is, how can we make sure that the MLP can predict correctly?

Our insight is to convert this problem into an image-based supervised training task by simply setting the aperture size as zero. In this condition, the composite cone will degrade to a cone (i.e., thin lens imaging will degrade to pinhole imaging). The rendered color of a pixel should be consistent with the ground truth

(GT) pixel color. If we scale up the pixel's receptive circle, the rendering result should be equivalent to the color of GT Pixel diffused by the same scale. This diffusion can be simply implemented using a Gaussian blur kernel. Further, if we introduce different scales for training, then different scales of each frustum will be involved, which constitutes a simple solution to supervise the MLP for correct prediction.

Consequently, we propose an efficient probabilistic training (p-training) strategy that largely simplifies the training process. The aperture size is set to zero during training. Each training step consists of the following: (1) Randomly choose a scale by predefined probabilities. (2) Scale the receptive circle of each pixel, calculate the composite cones, and divide them into frustums. (3) Calculate each frustum's expected PE and feed them into MLP to predict the density and radiance that represent the frustum. (4) Apply volume rendering on each composite cone to generate pixel colors. (5) Calculate the loss between the rendered colors and the colors of blurred GT. (6) Backpropagate the gradient decent to optimize the MLP parameters.

Although our p-training only applied on special cases (aperture size as zero), experiments show that the trained MLP can well generalize to various cases. At test time, NeRFocus can achieve high-quality defocus effects with adjustable camera pose, focus distance, or aperture size, e.g., by setting the aperture diameter as zero to render large DOF images as NeRF [21] does.

Despite such merits, NeRFocus incurs almost no additional cost on computation or parameters, neither sacrificing performance in rendering large DOF images.

In short, our contributions include:

- NeRFocus, an novel NeRF-based framework that implements thin lens imaging, by which we can render controllable 3D defocus effects.
- P-training, a probabilistic training strategy that eliminates the requirement of various depth-of-field datasets and vastly simplifies the training process.

2 Related Work

2.1 Neural Radiance Field

Neural Radiance Field (NeRF) [21] is one of the exciting pieces of work that has emerged in recent years at the intersection of neural networks and computer graphics. Previously, researchers have explored the potential of MLP to represent graphic properties implicitly [7], [31], [34], [10]. Volumetric representations [15], [25] and volume rendering [11], [18] also demonstrated the power of rendering high-quality novel views. Further, Mildenhall et al. proposed NeRF [21], which uses an MLP to represent a continuous volumetric scene function. By explicitly designing the view-dependent effects and using positional encoding, NeRF significantly improved the performance for rendering realistic complex scenes.

Methods has been proposed to further extend the performances of NeRF, e.g., accelerate training or inference time [23], [36], [9], [41], [16], improve the generalization performance [43], [38], [40], [4], enable dynamic scene rendering [29], [13],

[42], use unstructured images for training [17], [27], extend NeRF for multiscale and unbounded secens [23], [2], [3], scene disentanglement and control [44], [26], [24], [28], etc. This paper proposes the first framework that extends NeRF to approximate thin lens imaging, so the defocus effects can be rendered.

Our method is directly inspired by mip-NeRF [2], a multiscale representation for anti-aliasing rendering. Mip-NeRF extends the pixel's receptive field to a circle. Instead of sampling rays, mip-NeRF sampling conical frustums and use the Integrated Positional Encoding (IPE) to make the encoded frequency components reflect not only the changes of position but also the variance of frustum size. However, mip-NeRF's modeling is based on pinhole imaging, thus can not render lens effects. We explicitly model the thin lens imaging and derive composite cones that can be used to render each pixel into equivalent lens effects.

2.2 Synthetic Defocus Effects

In general, high-quality defocus effects heavily rely on expensive wide-aperture lenses. Thanks to advances in machine learning, some approaches have emerged to synthesize defocus effects from single or multiple large DOF images that mobile cameras can easily obtain. A simple solution for synthetic defocus is using a segmentation network [8], [5] to separate the foreground and background, then applying blur operation on the segmented background [32], [33], [39]. More general approaches are based on multiplane representations. For instance, one can apply depth estimation [14], [30] to divide the scene into several layers of different depths, then use depth-dependent kernels to blur these layers and composite them as an image to synthesize defocus effects [22], [45]. Similar methods can be applied to dual-lens images where the predicted disparity plays the role of depth [1], [39]. These image-based methods simplify the defocus effects in lens imaging as the "gather" operation on discretized depth layers, i.e., each pixel gathers influences from nearby pixels to form its color, which is usually implemented by convolution with a circular blur kernel. "Gather" operation may suffer edge artifacts if the depth or disparity estimation is inaccurate. Besides, the multiplane representation is hard to handle the occluded area between its discretized depth layers, making it difficult to synthesize free-view 3D defocus effects.

Recently, Mildenhall et al. [19] proposed RawNeRF and integrated it with the multiplane representation to synthesize 3D defocus effects. By training on raw images, RawNeRF can render images with a high dynamic range (HDR) that can be further retouched, such as changing exposure, tone mapping, and focus. To synthesize defocus effects, they first precompute a multiplane representation using the trained RawNeRF, then use the mentioned layer-wise blur operation to synthesize 3D defocus effects. However, NeRF-based precomputation of multiplane representation is either time-consuming or memory-consuming. Instead, NeRFocus, based on precise optical modeling, can directly render effects without post-process. Besides, our method should be compatible with RawNeRF's "HDR training", which may help improve our visual quality of defocus effects, especially the saliency of defocused bright highlights.

3 Method

Although NeRF and its variants achieve excellent performance in novel view synthesis, they can not render defocus effects efficiently. As mentioned earlier in the Introduction, the major difficulties for NeRF to render defocus effects constitute two: (1) The inherent limitations of the pinhole imaging model. (2) The prediction for different sizes of frustum's radiance and density.

Before we articulate how we solve these difficulties, let's briefly review the theory basics in NeRF and thin lens imaging.

3.1 Rendering in Neural Radiance Field

The neural radiance field (NeRF) [21] is a field of radiance (i.e., view-dependent color) and density that is consistently represented by a multilayer perceptron (MLP). To calculate the color of a pixel on the sensor plane, NeRF projects a ray $\mathbf{r}$ from the pixel to the pinhole and passes it through the scene space, then samples the ray by interval δ_i, $i \in \{1, ..., N\}$ to form N sample points $\mathbf{x}_i$. Positional Encoding (PE) is applied on $\mathbf{r}$ and each sample point $\mathbf{x}_i$:

$$\begin{aligned} \gamma(\mathbf{x}_i) &= \left[sin(\mathbf{x}_i), cos(\mathbf{x}_i), ..., sin(2^{L-1}\mathbf{x}_i), cos(2^{L-1}\mathbf{x}_i)\right]^\top, \\ \gamma(\mathbf{r}) &= \left[sin(\mathbf{r}), cos(\mathbf{r}), ..., sin(2^{M-1}\mathbf{r}), cos(2^{M-1}\mathbf{r})\right]^\top, \end{aligned} \tag{1}$$

where $\mathbf{x}_i$ represents the 3D location (x_i, y_i, z_i), L and M are hyperparameters, denoting the number of frequency components in PE. Tancik et al. [37] reveal that the MLP is insensitive to high-frequency variance unless explicitly provided, which makes PE essential to the success of NeRF. The resulting frequency components $\gamma(\mathbf{x}_i)$ and $\gamma(\mathbf{r})$ will be fed into the MLP to predict the corresponding color $\mathbf{c}_i$ and density σ_i:

$$[\sigma_i, \mathbf{c}_i] = \mathrm{MLP}(\gamma(\mathbf{x}_i), \gamma(\mathbf{r})). \tag{2}$$

The predicted colors and densities will be used for discrete volume rendering [18] to generate the pixel color $\mathbf{C}(\mathbf{r})$:

$$\begin{aligned} \mathbf{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \\ where \quad T_i &= \exp(-\sum_{j=1} \sigma_j(\delta_j)) \end{aligned} \tag{3}$$

The same rendering process is applied on every pixel of the sensor plane and finally composite a complete image.

Given a sparse set of posed images for training, NeRF chooses a pose for rendering and compares the result with the ground truth (GT) corresponding to the camera pose. Owing to the differentiable volume rendering formulation and the continuity of predicted color $\mathbf{c}_i$ and density σ_i, the gradient descent of loss function can smoothly backpropagate to the MLP parameters, thus making the MLP easy to converge to a plausible scene representation. In this way, NeRF achieves novel view synthesis without depth supervision.

3.2 Defocus in Thin Lens Imaging

The thin lens model ignores optical effects due to lens thickness and is usually used to simplify the analysis of lens imaging. The part (a) in Fig.2 illustrates a thin lens model. Given a thin lens with aperture diameter A, focal length f, and focus distance l, we can calculate the image distance l' following the Gaussian lens equation:

$$l' = \frac{fl}{f + l}. \tag{4}$$

Note that all "distance" in this paper denotes the distance along the optical axis, i.e., the z-axis by our definition. The focal plane is at the focus distance l and is orthogonal to the z-axis. The sensor plane is also orthogonal to the z-axis but located at the image distance l'. Here we take a point e in the scene space for analysis. If e is away from the focal plane, its imaging on the sensor plane is a circle rather than a point. This circle is called the circle of confusion (CoC) and is the immediate cause for defocus effects. Given e at distance z_e, we can get its image distance z'_e by Gaussian lens equation and use the properties of similar triangles to calculate its CoC diameter d_e:

$$z'_e = \frac{fz_e}{z_e + f} \quad with \quad \frac{A}{d_e} = \frac{z'_e}{|z'_e - l'|}, \quad i.e., \quad d_e = \left| \frac{Af(z_e - l)}{z_e(f + l)} \right|. \tag{5}$$

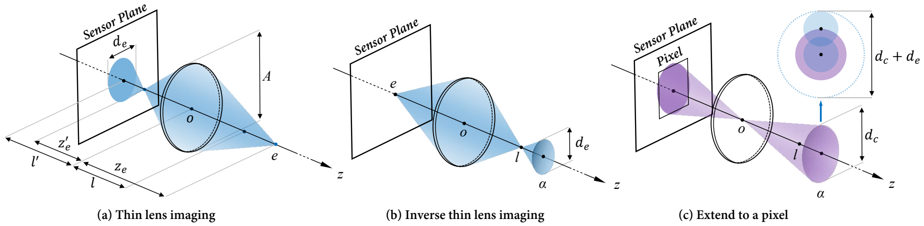
If we move e on the plane "z=z_e", its CoC diameter d_e is invariant. This attribute is critical for the following derivation.

3.3 Approximate Thin Lens Imaging in NeRF

Existing NeRFs are based on a pinhole imaging model, where the radiance of each object point in scene space is directly passed through the pinhole and imaging on the sensor plane as a point, without any lens-like optical refraction. Therefore, the whole scene is in sharp focus and defocus effects can not be rendered.

A typical way to approximate thin lens imaging in ray tracing is to sample enough rays and calculate the average returned radiance. But this will be computational expensive for NeRF, as the rendering of each ray needs dense predictions using MLP. Instead, we inverse the thin lens imaging to calculate the beam path of each pixel and directly render them. Fig.2 and Fig.3 illustrate our modeling.

Let's put the point e on the sensor plane for analysis. Following the same assumption in NeRF that the ray path is reversible, we project rays from e to every point on the lens plane. Actually, the corresponding refracted beam path constitutes a bicone that can be described in closed form following the lens equation. Say we have a plane "z=z_α" denoted as α, the bicone's cross-section on α should be a perfect circle, whose diameter d_e can be calculated following Eq.5, but with different parameters (symmetrically, if take e as the scene point and α to be the sensor plane, this circle can be seen as the CoC of e on the plane



(a) Thin lens imaging (b) Inverse thin lens imaging (c) Extend to a pixel

Fig. 2. Illustration for modeling thin lens imaging in NeRF. Note all "distance" in this paper denote the distance along the optical axis, i.e., z-axis, by our definition. (a) Defocus in thin lens imaging. If a point is away from the focal plane, its imaging on the sensor plane is a circle rather than a point, called the circle of confusion (CoC). (b) Inverse the thin lens imaging to form a bicone for each point on the sensor plane. The bicone can be seen as the beam path of a point's imaging. Note the ray from the point to the lens center is the axis of the bicone. α denotes the plane "$z = z_\alpha$". d_e denotes the bicone's diameter on plane α. (c) Use a circle to represent the receptive field of a pixel. The pixel's beam path forms a composite cone, which is overlapped by the beam paths of the points inside the receptive circle. As mentioned earlier, each point's beam path forms a bicone. Note the bicone's axis should cross the lens center, so all the axis of the bicones belonging to this pixel forms a cone (purple region), with d_c denoting its diameter on plane α. Then we expand the bicones along their axes, so the composite cone's diameter on plane α is exactly the sum of d_c and d_e. Fig.3 further illustrates the rendering process of the composite cone.

α):

$$d_e = \left| \frac{A(l - z_\alpha)}{l} \right| . \tag{6}$$

Ideally, we can get the point e's color by calculating the radiance returned by the bicone. But, how to get the color of a pixel? NeRF uses the color of a point to represent the color of a pixel, while in Mip-NeRF [2], Barron et al. extend the receptive field of a pixel as a circle, which achieves better rendering quality. Our method can easily incorporate this extension, as is shown in the (c) part of Fig.2. Say we have a pixel, and the point e is located at the pixel center. We use a circle centered on e, with diameter as d_0, to represent the receptive field of the pixel. Follow the practice in mip-NeRF, we set d_0 to the width of the pixel scaled by $2/\sqrt{3}$ for all experiments. The pixel's beam path forms a composite cone, which is overlapped by the beam paths of the points inside the receptive circle. We project lines from each point inside the circle through the lens center. Each line can be seen as the axis of a bicone. These lines constitute a cone (purple region in Fig.2). Based on parallel similarity, the cone's cross-section on α plane should always be a perfect circle, with diameter:

$$d_c = \frac{d_0 z_\alpha}{l'} = \frac{d_0 z_\alpha (f + l)}{fl} . \tag{7}$$

Let's take the cross-section on α plane for analysis. Since the cone's cross-section is a circle with diameter d_c, and each bicone's cross-section is a circle

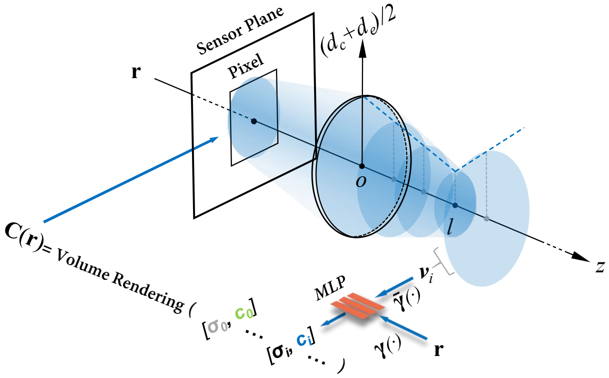


Fig. 3. Rendering process of a pixel. The line through the pixel center to the lens center is denoted as r, which can uniquely identify the composite cone of the pixel. The blue dotted line denotes the radius. We divide the composite cone into frustums, and use the modified integrated positional encoding (IPE), denoted as $\bar{\gamma}$, to encode each frustum $\mathbf{v}_i$. Then use an MLP to directly predict the expected radiance and density of each frustum. By volume rendering the predicted results, we can take the expected returned radiance of this composite cone as its corresponding pixel color.

with diameter d_e, the bicones overlap as a composite cone, whose diameter on α plane is:

$$d = d_c + d_e = \frac{d_0 z_\alpha (f + l)}{fl} + \left| \frac{A(l - z_\alpha)}{l} \right|, \tag{8}$$

where we can see that four parameters fully characterize the composite cone: pixel's receptive diameter d_0, aperture diameter A, focal length f, and focus distance l. Then we can get the pixel's color by calculating the composite cone's returned radiance. To simplify the computation, we partition the composite cone by putting the α plane on different distance intervals z_i, $i \in \{1, ..., N+1\}$ to form N conical frustums $\mathbf{v}_i$. We want to calculate a radiance $\mathbf{c}_i$ and density σ_i to represent each frustum $\mathbf{v}_i$, so we can use Eq.3 to calculate the returned radiance. But the problem is, how should we calculate the representative radiance and density?

In theory, every point inside the frustum may contribute radiance, but their "importance" may not equal, as the bicones in the composite cone overlap unevenly. Intuitively, a point close to the composite cone axis should contribute more because more bicone is overlapping. Therefore, we model the importance distribution inside frustum $\mathbf{v}_i$ as a 3D Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, with independent distribution of each coordinate x, y, z. Theoretically, our frustum's cross-section is parallel to the sensor plane while mip-NeRF's is orthogonal to the cone axis. But these differences bring negligible varies for the encoded value of frustum. So we adopt Mip-NeRF's coordinate system and it's IPE for simplicity. The mean value vector $\boldsymbol{\mu}_i$ is composed of (μ_x, μ_y, μ_z), the same value as the frustum's centroid coordinate. The covariance matrix $\boldsymbol{\Sigma}_i$ is a diagonal matrix with elements $\sigma_x, \sigma_y, \sigma_z$, i.e., the variances along each coordinate. These variances are set as the same value as the frustum's uniform distribution variance along each axis.

Therefore, we define the representative radiance $\mathbf{c}_i$ and density σ_i of a frustum $\mathbf{v}_i$ by calculating the expectation of radiance and density of every point inside this frustum:

$$[\sigma_i, \mathbf{c}_i] = \mathrm{E}_{\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}[\mathrm{MLP}(\gamma(\mathbf{x}), \gamma(\mathbf{r}))]. \tag{9}$$

A direct way to implement this operation is sampling enough points inside the frustum $\mathbf{v}_i$ follow the 3D Gaussian, using an MLP to predict their radiance and density, then calculate the average radiance and density. But this is extremely time-consuming. Inspired by Mip-NeRF, we use an MLP to directly predict the radiance of frustum $\mathbf{v}_i$, with the expected value of $\gamma(\mathbf{x})$ as input.

Specifically, each dimension x, y, z of the 3D Gaussian is independent of each other. Moreover, each frequency component in positional encoding (PE) is independent of each other as well, thus the distribution of each PE component can be simplified as a 1D Gaussian distribution. Say we have a coordinate x with distribution $\mathcal{N}(\mu_x, \sigma_x)$, the expected value of a frequency component $sin(2^k x), k \in \{0, ..., L-1\}$, in PE $\gamma(x)$ can be formulated as:

$$\mathrm{E}_{x \sim \mathcal{N}(\mu_x, \sigma_x)}[sin(2^k x)] = sin(2^k \mu_x) \exp(-(2^k \sigma_x)^2/2), \tag{10}$$

which can be easily promoted to each frequency component of each coordinate. In short, the expected value of $\gamma(\mathbf{x})$ inside frustum $\mathbf{v}_i$ can be easily calculated in closed form, denoted as:

$$\bar{\gamma}(\mathbf{v}_i) = \mathrm{E}_{\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}[\gamma(\mathbf{x})]. \tag{11}$$

Then we use the MLP to predict the radiance and density of frustum $\mathbf{v}_i$, with $\bar{\gamma}(\mathbf{v}_i)$ and $\gamma(\mathbf{r})$ as inputs:

$$[\sigma_i, \mathbf{c}_i] = \mathrm{MLP}(\bar{\gamma}(\mathbf{v}_i), \gamma(\mathbf{r})). \tag{12}$$

By volume rendering the predicted colors and densities of all frustums using Eq.3, we can calculate the radiance returned by a composite cone and use it to represent the corresponding pixel color. Fig.3 illustrates the rendering process of a pixel. The above formulation can be extended to every pixel on the sensor plane and finally approximate the optical effects of thin lens imaging.

3.4 P-Training Strategy

So far, we have built concise formulations to implement thin lens imaging in NeRF, but here comes another question: how can we make sure that the MLP can correctly predict the color and density for various sizes of frustum?

A direct solution is to collect enough images taken by a DSLR camera with a zoom lens and apply supervised training [35]. However, this solution is complex and demands extra datasets. Instead, we generate training datasets based on simple blur operations on the original dataset and apply a probabilistic training strategy based on the multiple blurred datasets. We name our training framework the p-training.

We consider a special case that the aperture is set to zero. Since $d_e = 0$, d is equal to d_c, i.e., the composite cone is degrade to a cone. Following Eq.7, d_c is proportional to d_0. If we scale d_0 using scalar k, we are essentially using the scaled composite cone to calculate the expected radiance as the pixel's color. Meanwhile, suppose we use the same scaled Gaussian to calculate the expectation of the pixel's color on the GT image. In that case, this color should be equal to the rendering color of the composite cone if the MLP can predict correctly by our definition in Eq.12 and Eq.9. Thus we obtain a simple equivalence between rendered results and images. Further, if we introduce different scales of k during training, different scales of composite cones will be involved, which constitutes a simple solution to supervise the MLP for correct prediction.

Specifically, given an original dataset $\mathbf{D} = \{\mathbf{y}_1, ..., \mathbf{y}_u\}$, where $\mathbf{y}$ denotes the images inside the dataset, we use different sizes of Gaussian blur kernel $\mathbf{k}_j$, $j \in \{1, ..., m\}$ to apply convolution operation (denoted as $*$) on every image in dataset $\mathbf{D}$:

$$\mathbf{D}_j = \{\mathbf{y}_1 * \mathbf{k}_j, ..., \mathbf{y}_u * \mathbf{k}_j\} \quad j \in \{1, ..., m\}, \tag{13}$$

which forms a set of multi-blur datasets $\{\mathbf{D}_1, ..., \mathbf{D}_m\}$.

As is shown in Fig.4, $\mathbf{D}_j \in \{\mathbf{D}_1, ..., \mathbf{D}_m\}$ is randomly chosen with predefined probabilities for each training setp (usually, give datasets with smaller $\mathbf{k}_j$ the higher chances to be chosen). We use the size of $\mathbf{k}_j$ to scale the composite cone's diameter. The rendering process is the same as is described before. Following the practice in mip-NeRF[2], we use a single MLP to implement a two-step coarse-to-fine training. For each composite cone, we first apply a uniform sampling $\mathbf{t}^c$ to divide frustums and calculate the "coarse" rendered color $\mathbf{C}(\mathbf{r}; \Theta, \mathbf{t}^c)$ and the importance distribution along the composite cone's axis, Then apply the importance sampling $\mathbf{t}^f$ to divide frustums and recalculate the "fine" rendered color $\mathbf{C}(\mathbf{r}; \Theta, \mathbf{t}^f)$. The training objective is to minimize the disparity between the blurred GT $\bar{\mathbf{C}}(\mathbf{r})$ and the composite cone's rendered color:

$$\min_{\Theta} \sum_{\mathbf{r} \in \mathcal{B}} (\lambda \left\| \bar{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}; \Theta, \mathbf{t}^c) \right\|_2^2 + \|\bar{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}; \Theta, \mathbf{t}^f)\|_2^2), \tag{14}$$

where Θ denotes parameters of the MLP. $\mathcal{B}$ denotes the ray batch. $\bar{\mathbf{C}}(\mathbf{r})$ is the pixel color in the dataset $\mathbf{D}_j$, i.e., the blurred pixel color using Gaussian kernel $\mathbf{k}_j$. $\mathbf{t}^c$ and $\mathbf{t}^f$ consist of 128 sampling points, respectively. λ is a hyperparameter.

Although our p-training only applied on special cases (aperture diameter as zero and scaled d_0), experiments show that the trained MLP can generalize to various lens parameters. We guess that this is largely related to the characteristics of IPE. Let's take Eq.10 for example, where $sin(2^k \mu_x)$ can be seen as the original PE. The essence of IPE lies on the coefficient $\exp(-(2^k \sigma_x)^2/2)$, which regulates the IPE's value range. If x has a more scattered distribution, i.e., high σ_x, the coefficient $\exp(-(2^k \sigma_x)^2/2)$ will approach zero. If the distribution is

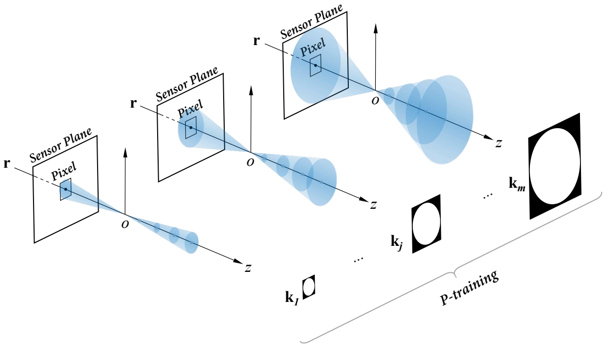


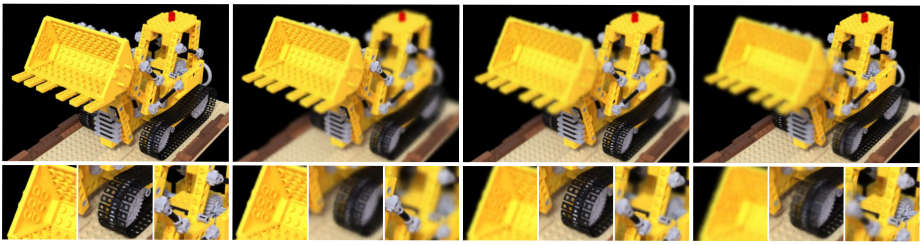
Fig. 4. Illustration of our p-training strategy. The aperture diameter is set to zero during training, so d_e is equal to zero, the composite cone is degrade as a cone. For each training step, we use predefined probabilities to select a blur kernel $\mathbf{k}_j$, $j \in \{1, ..., m\}$ randomly. Then, use the diameter of $\mathbf{k}_j$ to scale up every composite cone's diameters and use $\mathbf{k}_j$ to blur the original image as the rendering target. This training process will urge the MLP to correctly predict the radiance and density of frustums in different sizes.

fixed, a component with a higher frequency k will have a smaller value range. Therefore, $\mathrm{E}_{\mathbf{x}\sim\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)}[\cdot]$ can be seen as a low pass filter for $\gamma(\mathbf{x})$, considering $\gamma(\mathbf{x})$'s different frequency components. Meanwhile, applying $\mathrm{E}_{\mathbf{x}\sim\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)}[\cdot]$ on an image or on scene space will smooth the colors or $[\sigma_i, \mathbf{c}_i]$ pairs. Hence, the training for MLP to satisfy constraint $\mathrm{MLP}(\mathrm{E}[\gamma(\mathbf{x})], \gamma(\mathbf{r})) = \mathrm{E}[\mathrm{MLP}(\gamma(\mathbf{x}), \gamma(\mathbf{r}))]$ essentially lets the MLP learn such mapping: let the low-frequency components in IPE map to the blurred color, and gradually approximate the original color as the high-frequency components emerge. This type of mapping is easy for MLP to learn, as is proved in Fig.8.

Consequently, we can use the trained MLP to correctly predict colors and densities given frustums with different sizes and positions as inputs. By which we can render variant defocus effects. For instance, change the focus distance l to move the DOF forward or backward, change the aperture diameter A to raise or reduce the level of defocus effects.

4 Experiments

We evaluate NeRFocus on two types of dataset: the synthetic lego and materials dataset presented in NeRF [21] and the camera-captured horns and flower dataset presented in LLFF [20]. We set hyperparameters m=6, L=16, M=4 for all experiments. The scales of the blur kernels $\mathbf{k}_j \in \{1, ..., 6\}$ are set as $\{1, 3, 7, 15, 31, 51\}$ in units of pixel length, with probabilities $\{0.3, 0.2, 0.2, 0.1, 0.1, 0.1\}$ respectively. Note we use the original dataset when choosing kernel size 1. For training, we use the Adam optimizer with a logarithmically annealed learning rate that decays from 5×10^{-4} to 5×10^{-5}. The batch size of the composite cones



(a) A=0

(b) A=0. 06, l=3.4

(c) A=0.06, l=3.7

(d) A=0.06, l=4.0

Fig. 5. NeRFocus on the lego dataset. The camera pose is fixed for better comparison. We set the aperture diameter A as zero to synthesize image (a) where the whole scene is in sharp focus. We set A=0.06, l=3.4 to synthesize image (b) where the shovel is in sharp focus. We increase l to 3.7 to move the focus to the caterpillar band, as is shown in (c). We further increase l to 3.7 to move the focus to the gear wheel, as (d) shows. The results imply that our optical modeling has similar properties as a physical lens system.

is set as 4096 with 600000 training steps (higher training steps may elevate the PSNR but undermine the generalization performance in rendering defocus effects) on a single NVIDIA V100 GPU. For rendering during test time, we fix focal length f as 0.1, so the controllable parameters are aperture diameter A, focus distance l, and camera pose. This provides simple control for rendering various 3D defocus effects. Theoretically, changing l will result in changing l', which means changing the relative distance from the sensor plane to the lens center, so that all the composite cone's directions have to be recalculated, causing inconvenience for implementation. But since the defocus effects are independent to d_c, we simply set l' as constant to ignore the change of l' caused by the change of l.

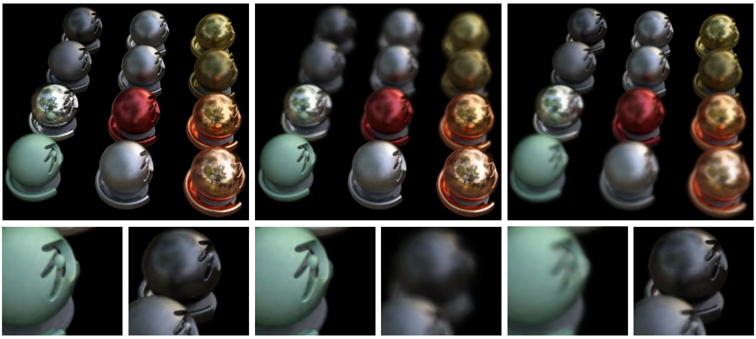
4.1 Quantitative Evaluation

Fig.8 visualizes the training process on the horns dataset (images are resized to 1008×756). Table 1 shows quantitative comparisons.

Though our approach can render various defocus effects, it incurs no additional costs on computation or parameters. Moreover, by simply setting the aperture A as zero, NeRFocus can render large DOF images as normal NeRF can do, with surprisingly close PSNR.

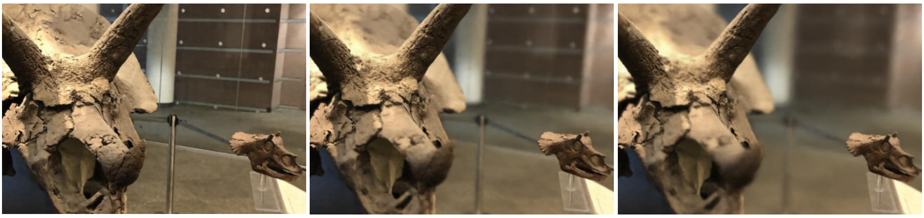
4.2 Qualitative Evaluation

Here we demonstrate various defocus effects that NeRFocus can render at test time. As is well known in photography, increasing the aperture diameter will narrow the DOF (i.e., enhance the defocus effects), while increasing the focus distance will move the DOF forward and vice versa. To verify that NeRFocus's rendering also conforms to these optical properties, we fix the camera pose and



(a) A=0 (b) A=0. 06, l=3.3 (c) A=0.06, l=4.6

Fig. 6. NeRFocus on the materials dataset. We set A as zero to synthesize image (a) where the whole scene is in sharp focus. We set A=0.06, l=3.3 to synthesize image (b) where the first row is in sharp focus. We increase l to 4.6 to move the focus to the last row, as (c) shows.



(a) A=0 (b) A=0.1 (c) A=0.2

Fig. 7. NeRFocus on the horns dataset. We fix the focus distance l at 0.68, and set A as 0, 0.1, 0.2 to synthesize images respectively. (a), (b), (c) shows the cropped results. We can see that the defocus effects get obvious when A increases, while the sharpness of the focused areas are not affected by larger aperture size. Notice that the blur varying along depth shows good continuity.

control the aperture diameter A and focus distance l respectively, to better compare the effects caused by A or l. Fig.5 and Fig.6 demonstrate the effects that varying focus distance l brings. Fig.7 shows the effects of changing aperture size A. We can see that NeRFocus is in good accordance with the mentioned optical properties, which indicates the effectiveness of our optical modeling. Besides, the blur varying along depth shows good continuity, which implies the excellent generalization performance of the MLP trained by our p-training strategy, especially considering that our p-training only uses five different blur kernels.

5 Discussion

We have presented NeRFocus, a framework that implements thin lens imaging in NeRF, by which we can achieve controllable 3D defocus effects. NeRFocus

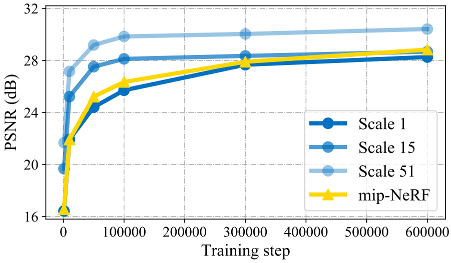


Fig. 8. PSNRs by training steps. The yellow line denotes mip-NeRF. The blue lines denote NeRFocus with different scales in p-training. Each scale's PSNR is calculated by the corresponding blurred GT. Note "scale 1" shares the same GT with mip-NeRF and achieves comparable performance to that of mip-NeRF, which indicates that each scale's learning is not contradictory but complementary.

Table 1. Quantitative comparisons on training time, model parameters, and PSNR. For NeRFocus, we set A=0 to calculate the PSNR by the original GT (large DOF). Though NeRFocus achieves defocus effects that mip-NeRF and NeRF cannot do, it neither incurs additional costs on computation and parameters nor sacrifices performance in rendering large DOF images.

Method	Time(hours)↓	Params↓	PSNR↑
NeRF (ECCV'20)	64.8	1192k	29.00
mip-NeRF (ICCV'21)	55.6	613k	28.83
NeRFocus (ours)	55.6	613k	28.26

does not need extra datasets with defocus effects. It only requires simple blur operations to generate its training dataset on the original large DOF dataset (e.g., videos taken by drones or cell phones). NeRFocus inherits NeRF's performance in rendering large DOF images and supports rendering shallow DOF images with flexible control on lens parameters. Though with such merits, the training and inference of NeRFocus do not bring extra consumption on time or parameters compared to NeRF or mip-NeRF. Our framework can be seen as a general extension of NeRF or mip-NeRF. e.g., set the aperture size A=zero, $(f+l)/fl$ as constant, and use a single kernel size $\{1\}$ for p-training, NeRFocus will degrade to mip-NeRF. We believe NeRFocus has great potential for broader applications and further improvements.

References

1. Barron, J.T., Adams, A., Shih, Y., Hernández, C.: Fast bilateral-space stereo for synthetic defocus. In: CVPR. pp. 4466–4474 (2015)
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. ICCV (2021)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. arXiv preprint arXiv:2111.12077 (2021)
4. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. arXiv preprint arXiv:2103.15595 (2021)
5. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. arXiv preprint arXiv:2112.01527 (2021)
6. Cook, R.L., Porter, T., Carpenter, L.: Distributed ray tracing. In: SIGGRAPH. pp. 137–145 (1984)
7. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH. pp. 303–312 (1996)
8. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017)
9. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. arXiv preprint arXiv:2103.14645 (2021)
10. Henzler, P., Mitra, N.J., Ritschel, T.: Learning a neural 3d texture space from 2d exemplars. In: CVPR. pp. 8356–8364 (2020)
11. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. ACM SIGGRAPH computer graphics **18**(3), 165–174 (1984)
12. Kolb, C., Mitchell, D., Hanrahan, P.: A realistic camera model for computer graphics. In: SIGGRAPH. pp. 317–324 (1995)
13. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR. pp. 6498–6508 (2021)
14. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. TPAMI **38**(10), 2024–2039 (2015)
15. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019)
16. Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., Saragih, J.: Mixture of volumetric primitives for efficient neural rendering. arXiv preprint arXiv:2103.01954 (2021)
17. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: CVPR. pp. 7210–7219 (2021)
18. Max, N.: Optical models for direct volume rendering. TVCG **1**(2), 99–108 (1995)
19. Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P., Barron, J.T.: Nerf in the dark: High dynamic range view synthesis from noisy raw images. arXiv preprint arXiv:2111.13679 (2021)

20. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
21. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421. Springer (2020)
22. Narain, R., Albert, R.A., Bulbul, A., Ward, G.J., Banks, M.S., O'Brien, J.F.: Optimal presentation of imagery with focus cues on multi-plane displays. ACM Transactions on Graphics (TOG) **34**(4), 1–12 (2015)
23. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. arXiv preprint arXiv:2103.03231 (2021)
24. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: CVPR. pp. 11453–11464 (2021)
25. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: CVPR. pp. 3504–3515 (2020)
26. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: CVPR. pp. 2856–2865 (2021)
27. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV. pp. 5865–5874 (2021)
28. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. In: SIGGRAPH Asia (2021)
29. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: CVPR. pp. 10318–10327 (2021)
30. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. TPAMI (2019)
31. Ren, P., Wang, J., Gong, M., Lin, S., Tong, X., Guo, B.: Global illumination with radiance regression functions. ACM Transactions on Graphics (TOG) **32**(4), 1–12 (2013)
32. Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E., Sachs, I.: Automatic portrait segmentation for image stylization. In: Computer Graphics Forum. vol. 35, pp. 93–102. Wiley Online Library (2016)
33. Shen, X., Tao, X., Gao, H., Zhou, C., Jia, J.: Deep automatic portrait matting. In: European conference on computer vision. pp. 92–107. Springer (2016)
34. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. arXiv preprint arXiv:1906.01618 (2019)
35. Srinivasan, P.P., Garg, R., Wadhwa, N., Ng, R., Barron, J.T.: Aperture supervision for monocular depth estimation. In: CVPR. pp. 6393–6401 (2018)
36. Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P.P., Barron, J.T., Ng, R.: Learned initializations for optimizing coordinate-based neural representations. In: CVPR. pp. 2846–2855 (2021)
37. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)

38. Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d representation and rendering. In: ICCV. pp. 15182–15192 (2021)
39. Wadhwa, N., Garg, R., Jacobs, D.E., Feldman, B.E., Kanazawa, N., Carroll, R., Movshovitz-Attias, Y., Barron, J.T., Pritch, Y., Levoy, M.: Synthetic depth-of-field with a single-camera mobile phone. ACM Transactions on Graphics (ToG) **37**(4), 1–13 (2018)
40. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR. pp. 4690–4699 (2021)
41. Wang, Z., Bagautdinov, T., Lombardi, S., Simon, T., Saragih, J., Hodgins, J., Zollhofer, M.: Learning compositional radiance fields of dynamic human heads. In: CVPR. pp. 5704–5713 (2021)
42. Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: CVPR. pp. 9421–9431 (2021)
43. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR. pp. 4578–4587 (2021)
44. Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., Yu, J.: Editable free-viewpoint video using a layered neural representation. ACM Transactions on Graphics (TOG) **40**(4), 1–18 (2021)
45. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. SIGGRAPH (2018)