PDRF: Progressively Deblurring Radiance Field for Fast Scene Reconstruction from Blurry Images

Cheng Peng and Rama Chellappa

Johns Hopkins University
cpeng26@jhu.edu

Abstract

We present Progressively Deblurring Radiance Field (PDRF), a novel approach to efficiently reconstruct high quality radiance fields from blurry images. While current State-of-The-Art (SoTA) scene reconstruction methods achieve photo-realistic renderings from clean source views, their performances suffer when the source views are affected by blur, which is commonly observed in the wild. Previous deblurring methods either do not account for 3D geometry, or are computationally intense. To addresses these issues, PDRF uses a progressively deblurring scheme for radiance field modeling, which can accurately model blur with 3D scene context. PDRF further uses an efficient importance sampling scheme that results in fast scene optimization. We perform extensive experiments and show that PDRF is **15X faster** than previous SoTA while achieving **better performance** on both synthetic and real scenes.

Introduction

Reconstructing a 3D scene from 2D images is a long-standing research problem with extensive applications in robotics, site model construction, AR/VR, e-commerce, etc. Significant progress has been observed recently with developments in differentiable rendering and implicit function representation, e.g. Neural Radiance Field, or NeRF (Mildenhall et al. 2020). In this approach, scene radiance is represented as a neural network, i.e. a Multi-Layer Perceptron (MLP), which maps 3D coordinates to their respective density and color. By incorporating the MLP into ray tracing, scene radiance is optimized with a self-supervised photometric loss to satisfy multi-view observations; therefore, only calibrated images are required. This formulation produces photo-realistic scene rendering, models scene geometry, and accounts for view dependent effects.

While NeRF, along with other implicit function-based methods (Park et al. 2019; Mescheder et al. 2019; Chen and Zhang 2019; Sitzmann, Zollhöfer, and Wetzstein 2019), marks a significant step towards photo-realistic scene reconstruction, its application in the real world is still limited. Images acquired in the wild are often affected by factors like object motion, camera motion, incorrect focus, or low
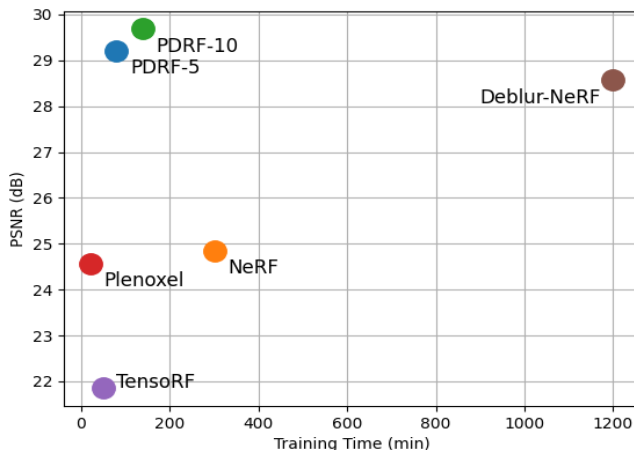
Figure 1: Comparisons of radiance field modeling methods from *blurry* images. Our proposed PDRF significantly outperforms previous methods in speed and performance. Results are aggregated from Table 1.

light conditions. These factors can significantly impact radiance field modeling due to information loss and multi-view *in*consistency. There often exists enough information within images in the wild to model high quality radiance field, despite the sub-optimal acquisitions. To this end, subsequent works (Mildenhall et al. 2022; Park et al. 2021a,b; Pumarola et al. 2021; Tretschk et al. 2021; Martin-Brualla et al. 2021) attempt to account for these conditions with domain knowledge and estimate a robust NeRF.

In this work, we seek to better address camera motion and defocus blur in radiance field modeling, which is commonly observed. Prior works have two drawbacks: a lack of 3D context in modeling blur and a time-consuming optimization process. In particular, Deblur-NeRF (Ma et al. 2022) models a blurry observation by blending P clean pixels with a blur kernel; however, the pixel-wise blur is predicted based on 2D image coordinates. As we will show, this limits Deblur-NeRF's blur modeling capacity due to the lack of scene context. Since P pixels need to be ray-traced to render a single blurry pixel, Deblur-NeRF is also P times slower than NeRF in rendering the same number of pixels. As shown in Fig. 1, this make Deblur-NeRF very computationally demanding.

Several previous works (Fridovich-Keil et al. 2022; Müller et al. 2022; Sun et al. 2022; Chen et al. 2022) have shown great progress in accelerating NeRF; however, they are not directly applicable to methods that dynamically generated rays to account for view inconsistencies. Specifically, these acceleration methods rely on an explicit density volume, which allows fast ray-tracing by pruning away low density voxels. This strategy relies on the assumption that static rays can lead to good density approximation, which is not true when images are blurred. Voxel pruning also leads to sub-optimal learning for dynamic ray generation because gradients only propagate on parts of the ray.

We propose a novel Progressively Deblurring Radiance Field, or PDRF. PDRF uses a Progressive Blur Estimation (PBE) module to address the lack of 3D modeling for blur. To achieve this, PBE first generates a coarse set of rays to account for blur, based on 2D coordinates. PBE then traces these rays and obtains their features from the underlying radiance field. Finally, PBE refines coarse blur estimations with a second update by incorporating 3D features into blur modeling. For rendering, PDRF uses a Coarse Ray Renderer (CRR) and a Fine Voxel Renderer (FVR). CRR provides feature and density estimation for PBE and FVR respectively. In particular, CRR achieves efficiency by rendering a ray from its aggregated feature instead of rendering individual voxels. FVR focuses on quality, and uses a larger network and denser predictions for high quality view synthesis. CRR and FVR combine the use of explicit features (Chen et al. 2022) and an efficient importance sampling to enable fast convergence and proper gradient updates for PBE. As demonstrated in Fig. 1, despite the more sophisticated blur modeling, PDRF is faster *and* more performant than Deblur-NeRF in modeling radiance field from blurry images. In summary, our contributions can be described in three parts:

1. We propose a Progressive Blur Estimation module, which first proposes a coarse blur model and updates it with corresponding 3D features to capture details.

2. We propose an efficient rendering scheme, which includes a Coarse Ray Renderer and a Fine Voxel Renderer; this scheme is significantly faster than NeRF while supporting ray optimization in PBE.

3. We evaluate the overall PDRF method on synthetic and real images with blur, and find significant performance improvements compared to SoTA methods.

Related Work and Backgrounds

Motion and Defocus Deblurring. Motion and defocus blur are commonly observed image degradations. Blurry observations are typically modeled as convolutions between clean images and blur kernels; therefore, single image deblurring is a heavily ill-posed problem, as many solutions can lead to the same observations. Traditionally, various priors are used to constraint the solution space (Chan and Wong 1998; Rudin, Osher, and Fatemi 1992; Krishnan, Tay, and Fergus 2011; Xu, Zheng, and Jia 2013; Tai et al. 2011; Levin 2006). More recently, deep learning-based methods have achieved great performances by learning a more sophisticated deblurring prior in a data driven way, e.g. as a function that directly maps blurry images to clean images. (Chakrabarti 2016; Kupyn et al. 2018; Lee et al. 2021; Nah, Kim, and Lee 2017; Son et al. 2021a; Tao et al. 2018; Zamir et al. 2021; Zhang et al. 2020) Video-based deblurring explores temporal consistency to find better solutions, e.g. through optical flow (Kim and Lee 2015; Pan, Bai, and Tang 2020; Li et al. 2021) or recurrent structures (Zhong et al. 2021; Kim et al. 2017; Nah, Son, and Lee 2019; Son et al. 2021b).

Neural Radiance Field Modeling. Significant progress has been made in recent years on the 3D scene reconstruction task. Due to space constraints, we refer to previous surveys (Han, Laga, and Bennamoun 2021; Tewari et al. 2020) for a comprehensive review. We focus on neural radiance field modeling, a SoTA approach for scene reconstruction. NeRF (Mildenhall et al. 2020) is a seminal work that models scene radiance with an implicit MLP function F. The color $\mathbf{c}_i$ and density σ_i at *voxel* location $X_i \in \mathbb{R}^3$ can be queried from F:

$$(\mathbf{c}_i, \sigma_i) = F(\gamma_{L_X}(X_i), \gamma_{L_d}(d)), \tag{1}$$

where $\mathbf{c}_i$ is dependent on view direction d, and γ_L is an L-frequency-band embedding function that leads to better convergence (Tancik et al. 2020). By tracing a ray at locations $X_i = o + t_i d$, where o is the ray origin and t_i indicates the travel distance, the rendered color $\hat{C}(\mathbf{r}_x)$ for ray $\mathbf{r}_x$ is:

$$
\begin{aligned}
\hat{C}(\mathbf{r}_x) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \\
T_i &= \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j),
\end{aligned}
\tag{2}
$$

which aggregate color $\mathbf{c}_i$ based current and cumulative density $\{\sigma_i, T_i\}$, modulated by distance between samples δ_i. We update F by constraining $\hat{C}(\mathbf{r}_x)$ to be similar to observations $C(\mathbf{r}_x)$ at all *pixel* locations $x \in \mathbb{R}^2$.

NeRF can produce photo-realistic renderings on novel views; however, it relies on a powerful MLP to represent the scene. This leads to a very long optimization time with millions of queries from a network. To ameliorate this issue, NeRF applies importance sampling: it first samples few equidistant locations along a ray before proposing additional points around high density regions for finer sampling. Many follow-up works seek to further accelerate NeRF optimization, generally following a space-time trade-off by introducing explicit representations to supplement implicit neural networks. DirectVoxGo (Sun et al. 2022) proposes to use feature and density volumes to store view-invariant information and uses a shallow MLP to render view-dependent color. Plenoxel (Fridovich-Keil et al. 2022) discards MLPs and uses spherical harmonics to model view dependent effects; the voxel density and spherical harmonic coefficients are stored as volumes. A downside to using volumetric representations is the large memory footprint. To this end, Instant-NGP (Müller et al. 2022) uses multi-resolution hashing instead of a volume to compactly encode explicit features. TensoRF (Chen et al. 2022) proposes a tensor decomposition to approximate 3D features in lower dimensions, which is more scalable to high resolution. Another
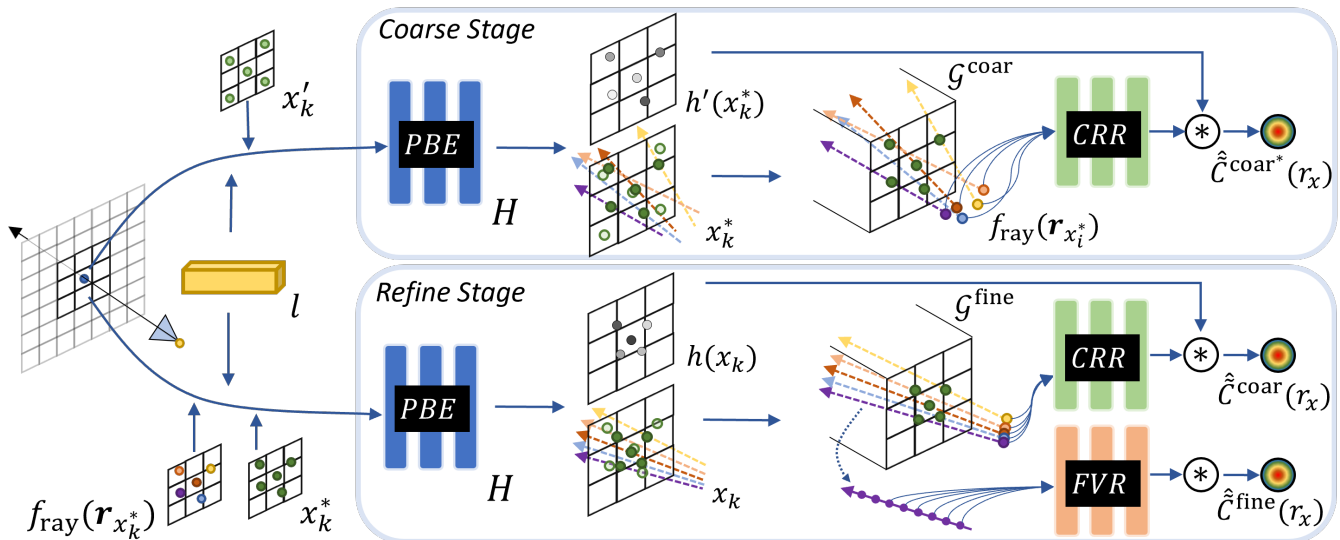
Figure 2: PDRF is split into two stages. In the coarse stage, PBE models the blur from 2D coordinates to obtain the initial kernel locations x_k^*. We then obtain the respective ray features $f_{\text{ray}}(\mathbf{r}_{x_k^*})$ from CRR. PBE then generates a second, refined blur estimate based on $\{x_k^*, f_{\text{ray}}(\mathbf{r}_{x_k^*})\}$, and uses CRR and FVR to render the observed blurry pixel. After optimization, we can directly render from a blur-free radiance field to obtain deblurred images.

approach to accelerate NeRF is through more efficient importance sampling. Mip-NeRF360 (Barron et al. 2022) uses a small coarse network that outputs density only and is supervised by another large network. PDRF combines a novel and efficient importance sampling scheme with explicit representations to realize greater acceleration.

NeRF Modeling With Non-Ideal Images. NeRF works well when images are clean and well-calibrated; however, images acquired in the wild are often non-ideal. Factors, such as low light, camera motion, object motion, and incorrect focus, can degrade image quality or affect multi-view consistency. Many works introduce domain knowledge to model the non-ideal contributor to observations, such that the radiance field function only models a canonical, multiview consistent scene. RawNeRF (Mildenhall et al. 2022) uses an approximated tonemapped loss and variable exposure to account for low light observations. NeRF-W (Martin-Brualla et al. 2021) uses image-specific embedding to model inconsistent appearances and transient objects. For object motion, many works (Park et al. 2021a,b; Pumarola et al. 2021; Tretschk et al. 2021) use an implicit function to describe scene deformation, such that the rays are first deformed, then traced by a radiance field function. Recently, Deblur-NeRF (Ma et al. 2022) introduces an implicit function to model per-pixel blur before radiance field modeling. Specifically, the blurry observation $\tilde{C}$ is modeled as a convolution between clean pixels and a blur kernel:

$$\tilde{C}(\mathbf{r}_x) = \sum_{x_k \in \mathcal{N}(x)} C(\mathbf{r}_{x_k}) h(x_k) \text{ s.t} \sum_{x_k \in \mathcal{N}(x)} h(x_k) = 1,$$
$$(3)$$

where x_k denotes the clean 2D coordinates at neighboring locations $\mathcal{N}(x)$. The implicit function accounts for blur by predicting $P = |\{x_k\}|$ rays and their blending weights

$h(x_k)$, which are used by a NeRF to computes $\tilde{C}(\mathbf{r}_{x_k})$.

Progressively Deblurring Radiance Field

As shown in Fig. 2, PDRF consists of two parts: a Progressive Blur Estimation (PBE) module, and a rendering module. We first describe PBE, which models blur in two stages. The coarse stage is fast and estimates blur based on 2D pixel coordinates; the finer stage updates previous results by adding ray features, such that PBE can take into account 3D scene context and model blur with more precision. We then describe the rendering module, which combines an efficient importance sampling scheme with explicit representations to accelerate optimization. Coarse Ray Renderer (CRR) provides fast ray feature and density estimations for PBE and Fast Voxel Renderer (FVR). FVR performs importance sampling based on CRR's density estimation and has finer ray tracing resolution; it also uses a larger, voxel-wise rendering network to produce high quality view synthesis.

Progressive Blur Estimation

We follow prior works by introducing an implicit function H to model blur. Specifically, H estimates a number of x_k and their blur kernel weights $h(x_k)$ for every x; we then can use Eq. (3) to render the blurry observations from x_k.

Since tracing multiple $\mathbf{r}_{x_k}$ is costly, the kernel size P should be minimal. To effectively find the P rays that describe blur, we propose a two-stage estimation scheme. As shown in Fig. 2, the coarse blur estimation stage makes use of a canonical blur kernel location $x_k' \in \mathbb{R}^{V \times P \times 2}$ and view embedding $l \in \mathbb{R}^{V \times K}$, where V is the number of source views and K is the embedding size. Both x_k' and l are learnable parameters that serve as the blur kernel and view index

input to H. In return, H produces a set of coarse ray kernels $\mathbf{r}_{x_k^*}$ to account for blurry observations in Eq. (3). More specifically, H produces offsets $\{\Delta o_{x_k'}, \Delta x_k', \}$ to the ray origin o and x_k'. This can be expressed as:

$$\Delta o_{x_k'}, \Delta x_k', h'(x_k^*) = H(l, x, x_k', \emptyset),$$
$$x_k^* = x_k' + \Delta x_k', o_{x_k^*} = o + \Delta o_{x_k'}, \qquad (4)$$
$$\mathbf{r}_{x_k^*} = o_{x_k^*} + t d_{x_k^*},$$

where $\emptyset$ is a zero vector placeholder for features from the next stage. This initial blur model $\{\mathbf{r}_{x_k^*}, h'(x_k^*)\}$ is similar in formulation to Deblur-NeRF (Ma et al. 2022) and has been shown to work reasonably well. As recent works (Fridovich-Keil et al. 2022; Sun et al. 2022; Müller et al. 2022; Chen et al. 2022) suggest, an implicit function can be enhanced by explicit features for better and faster convergence, we hypothesize that H can also benefit from additional features.

To motivate the correct features that help H account for blur, consider a complex scene in Fig. 3a where the background is out-of-focus. A key piece of information that describes blur in a pixel is its depth, as pixels have more blur when their content is further away from the camera. While $\{\mathbf{r}_{x_k^*}, h'(x_k^*)\}$ can model a coarse blurry regions, it cannot distinguish the sharp foreground details amongst background pixels, limited by their similar 2D coordinates. Based on this observation, we introduce a second, refinement stage that incorporates ray features f_{ray} to H for fine-grained blur modeling. As shown in Fig. 2, $f_{\text{ray}}(\mathbf{r}_{x_k^*})$ is obtained by tracing and aggregating per-voxel features from the underlying radiance field. The exact formulation is elaborated in Eq. (7) and the next section.

Based on $f_{\text{ray}}(\mathbf{r}_{x_k^*})$, PBE performs a second inference to update $\{\mathbf{r}_{x_k^*}, h'(x_k^*)\}$ and produces the final $\mathbf{r}_{x_k}$:

$$\Delta o_{x_k^*}, \Delta x_k^*, h(x_k) = H(l, x, x_k^*, f_{\text{ray}}(\mathbf{r}_{x_k^*})),$$
$$x_k = x_k^* + \Delta x_k^*, o_{x_k} = o + \Delta o_{x_k^*}, \qquad (5)$$
$$\mathbf{r}_{x_k} = o_{x_k} + t d_{x_k}.$$

The updated $\{\mathbf{r}_{x_k}, h'(x_k)\}$ is more accurate in describing blur, which we demonstrate in Fig. 3c and 3d by measuring the variances of $\Delta o_{x_k^*}$ and Δo_{x_k}. When observations are already sharp, i.e. a single ray can approximate the observation, $\{\Delta o, \Delta x'\}$ should have to similar spatial locations; when observations are blurry, $\{\Delta o, \Delta x'\}$ should be dissimilar to account for blur and have *high variance*. Clearly, $\{\mathbf{r}_{x_k}, h'(x_k)\}$ can better distinguish the clear foreground details from the blurry background. Furthermore, a two-stage design is still efficient since Eq. (4) computes quickly.

Coarse Ray Renderer

The Coarse Ray Renderer serves two roles in PDRF: it provides coarse density estimation to FVR and ray features to PBE. CRR accomplishes this by performing a fast ray tracing. To estimate voxel density, CRR uses a two-layer MLP $\mathcal{M}_\sigma^{\text{coar}}$ and an explicit feature volume $\mathcal{G}^{\text{coar}}$:

$$\sigma_i^{\text{coar}}, f_{X_i}^{\text{coar}} = \mathcal{M}_\sigma^{\text{coar}}(\gamma_{L_X}(X_i), \mathcal{G}^{\text{coar}}(X_i)), \qquad (6)$$

where X_i is the traced 3D coordinates. We implement $\mathcal{G}^{\text{coar}}$ with tensor decomposition (Chen et al. 2022) and

(a) Defocus Blur (b) Deblur Blur, Magnified

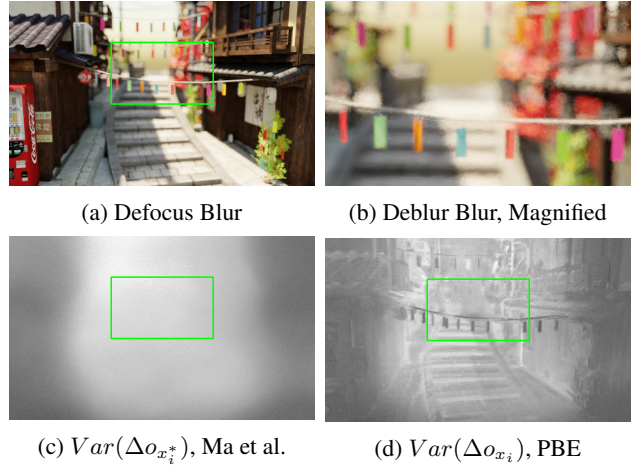(c) $Var(\Delta o_{x_i^*})$, Ma et al. (d) $Var(\Delta o_{x_i})$, PBE

Figure 3: Visualization of defocus blur in (a), background is out-of-focus. We visualize the variance of predicted Δo_{x_i}, where higher variance correlates with more blurry regions. Our PBE module (d) can clearly distinguish foreground details despite their proximity with background pixels; (c) can only coarsely distinguish fore/background.

use it to accelerate convergence for $\mathcal{M}_\sigma^{\text{coar}}$. We find that a purely explicit density representation used by previous methods (Fridovich-Keil et al. 2022; Sun et al. 2022; Chen et al. 2022) is prone to significant noise when the observations have complex details or blur, likely due to its inflexible volumetric data structure. Therefore, Eq. (6) uses a combination of an implicit function and explicit features to ensure good density estimation. CRR further obtains an efficient ray feature representation by modifying on Eq. (2):

$$f_{\text{ray}}(\mathbf{r}_x) = \sum_{i=1}^{N_c} T_i^{\text{coar}} (1 - \exp(-\sigma_i^{\text{coar}} \delta_i)) f_i^{\text{coar}}, \qquad (7)$$

where $f_{\text{ray}}(\mathbf{r}_x)$ aggregates feature along ray $\mathbf{r}_x$, similar to the aggregated color $C(\mathbf{r}_x)$, and N_c is the number of equidistant samples. We find $f_{\text{ray}}(\mathbf{r}_x)$ to be an effective 3D representation for PBE; in particular, $f_{\text{ray}}(\mathbf{r}_x)$ is agnostic to ray direction as the blur model in PBE is already view specific. We use $f_{\text{ray}}(\mathbf{r}_x)$ for color rendering by another MLP $\mathcal{M}_r^{\text{coar}}$:

$$\hat{C}^{\text{coar}}(\mathbf{r}_x) = \mathcal{M}_r^{\text{coar}}(f_{\text{ray}}(\mathbf{r}_x), \gamma_{L_d}(d)). \qquad (8)$$

This ray-rendering formulation has two advantages. First, it is very fast to compute; instead of querying every voxel along the ray through an MLP, Eq. (8) is only inferenced once per ray. It also serves as a coarse approximation of per-voxel rendering. Consider ray tracing when the surface density is a Dirac delta function $\sigma = \hat{\delta}(X_{\text{surf}})$ at a 3D location X_{surf}; this simplifies Eq. (7) and Eq. (8) to $\hat{C}^{\text{coar}}(\mathbf{r}) = \mathcal{M}_r^{\text{coar}}(f_{X_{\text{surf}}}^{\text{coar}}, \gamma_{L_d}(d))$, which is equivalent to per-voxel rendering. While this is the optimal solution, multi-view observations constrain f_{ray} to coarsely represent surface features and help CRR achieve a good balance between speed and performance. CRR is first used by PBE to gather ray features and renders color based on $\{\mathbf{r}_{x_k^*}, h'(x_k^*)\}$; it is then used with FVR to render colors based on the updated

$\{\mathbf{r}_{x_k}, h(x_k)\}$. Following Eq. (3), the predicted blurry colors are computed as follows:

$$\hat{\tilde{C}}^{\text{coar}^*}(\mathbf{r}_x) = \sum_{x_k^*} \hat{C}^{\text{coar}}(\mathbf{r}_{x_k^*}) h'(x_k^*),$$

$$\hat{\tilde{C}}(\mathbf{r}_x) = \sum_{x_k^*} \hat{C}(\mathbf{r}_{x_k}) h'(x_k), \qquad (9)$$

$$\hat{\tilde{C}}^{\text{coar}}(\mathbf{r}_x) = \sum_{x_k} \hat{C}^{\text{coar}}(\mathbf{r}_{x_k}) h(x_k).$$

Its training loss is a combined photometric loss with all blurry observation $\tilde{C}(\mathbf{r}_x)$:

$$\mathcal{L}_{photo}^{CRR} = \|\hat{\tilde{C}}^{\text{coar}^*}(\mathbf{r}_x) - \tilde{C}(\mathbf{r}_x)\|_2^2 + \|\hat{\tilde{C}}^{\text{coar}}(\mathbf{r}_x) - \tilde{C}(\mathbf{r}_x)\|_2^2. \qquad (10)$$

Fine Voxel Renderer

The Fine Voxel Renderer leverages the density estimation from CRR and generates high quality rendering. CRR calculates weights $w_i = T_i(1 - \exp(-\sigma_i^{\text{coar}}\delta_i))$ for N_c equidistant points in Eq. (7), these weights are normalized to produce a PDF of ray density. FVR uses this PDF to sample additional N_f points on high density regions for finer tracing. Importance sampling scheme is used instead of voxel pruning, as voxel pruning introduces optimization issues for PBE. Specifically, previous works (Fridovich-Keil et al. 2022; Sun et al. 2022; Chen et al. 2022) use explicit density volume and quickly prune off many low density voxels. While this works for *static* rays, PBE dynamically generates rays $\mathbf{r}_{x_i}$ based on gradients from the photometric loss. In this case, voxel pruning also eliminates gradients on many points and forces PBE to optimize $\mathbf{r}_{x_i}$ base on small segments of the ray. Furthermore, high density voxels are often not correctly estimated from blurry observations and can lead to erroneous pruning.

After importance sampling, FVR uses a large MLP, which contains density and color components $\mathcal{M}_\sigma^{\text{fine}}$ and $\mathcal{M}_c^{\text{fine}}$ respectively for high quality view synthesis. The voxel density σ^{fine} is obtained as follows:

$$\sigma^{\text{fine}}, f_{X_i}^{\text{fine}} = \mathcal{M}_\sigma^{\text{fine}}(\gamma_{L_v}(X_i), \mathcal{G}^{\text{coar}}(X_i) \oplus \mathcal{G}^{\text{fine}}(X_i)), \quad (11)$$

where $\oplus$ is the concatenation operation. We use an additional fine feature volume $\mathcal{G}^{\text{fine}}$ with the previous $\mathcal{G}^{\text{coar}}$ to maximize explicit feature capacity. Then, per-voxel color is rendered and aggregated as follows:

$$\mathbf{c}_i = \mathcal{M}_c^{\text{fine}}(f_{X_i}^{\text{fine}}, \gamma_{L_d}(d)),$$

$$\hat{C}^{\text{fine}}(\mathbf{r}_x) = \sum_{i=1}^{N_c+N_f} T_i^{\text{fine}}(1 - \exp(-\sigma_i^{\text{fine}}\delta_i))\mathbf{c}_i. \qquad (12)$$

A photometric loss is used for FVR, similar to CRR:

$$\hat{\tilde{C}}^{\text{fine}}(\mathbf{r}_x) = \sum_{x_i} \hat{C}^{\text{fine}}(\mathbf{r}_{x_i}) h(x_i),$$

$$\mathcal{L}_{photo}^{FVR} = \|\hat{\tilde{C}}^{\text{fine}}(\mathbf{r}_x) - \tilde{C}(\mathbf{r}_x)\|_2^2. \qquad (13)$$

Finally, the overall loss includes a total variation constraint $\mathcal{L}_{TV}^*$ on the explicit feature volumes to enforce smoothness,

following previous works (Fridovich-Keil et al. 2022; Sun et al. 2022; Chen et al. 2022):

$$\mathcal{L}_{tot} = \mathcal{L}_{photo}^{CRR} + \mathcal{L}_{photo}^{FVR} + \lambda(\mathcal{L}_{TV}^{\mathcal{G}^{\text{coar}}} + \mathcal{L}_{TV}^{\mathcal{G}^{\text{fine}}}). \qquad (14)$$

Experiments

Dataset. We evaluate our deblurring method on the dataset provided by (Ma et al. 2022), which contains five synthetic scenes and twenty real world scenes. The synthetic scenes are affected by camera motion blur and defocus blur separately and have blur-free novel view groundtruth. For real world scenes, ten of them are affected by camera motion blur and the remaining ten are affected by defocus blur. While real world scenes have blur-free references, the calibration and exposure may not be consistent with source views. As such, we report quantitative evaluations, specifically PSNR and SSIM, on synthetic scenes and provide quantitative metrics on real world scenes in supplemental material.

Implementation Details. We construct PBE as a four-layer MLP with a channel size of 64 and ReLU as the activation function. The size of view embedding l and ray feature f_{ray} is 32 and 15. The canonical kernel locations x_i' are normalized to be at most ± 10 pixels from x. For CRR, we use a decomposed feature tensor $\mathcal{G}^{\text{coar}}$ that represents 17 million voxels; for FVR, $\mathcal{G}^{\text{fine}}$ represents 134 million voxels. The channel dimension for the decomposed $\{X, Y, Z\}$ axes are $\{64, 16, 16\}$. For CRR, the two-layer MLP $\mathcal{M}_\sigma^{\text{coar}}$ and three-layer MLP $\mathcal{M}_r^{\text{coar}}$ have a channel size of 64 for hidden layers. PDRF is trained with a batch size of 1024 rays, each ray has coarse samples $N_c = 64$ and importance sampled for $N_f = 64$ additional voxels. A gamma correction function of $\hat{C}(\mathbf{r}_x)^{\frac{1}{2.2}}$ is used as the camera response function.

Ablation Study. We compare three versions of designs to demonstrate the effectiveness of different components:

- PDRF$^{\text{single}}$-5: We use only the first stage, i.e. Eq. (4), to model blur without ray features, kernel size $P = 5$.
- PDRF-5: PDRF with kernel size $P = 5$.
- PDRF-10: PDRF with kernel size $P = 10$.

As shown in Table 1, PDRF$^{\text{single}}$-5 is much faster than Deblur-NeRF and on par in performance, demonstrating the effectiveness of CRR and FVR; however, PDRF-5 outperforms PDRF$^{\text{single}}$-5 by introducing a second refinement stage in PBE. PDRF-5 is not significantly slower than PDRF$^{\text{single}}$-5, as CRR can obtain ray features very quickly. Finally, PDRF-10 uses a more powerful blur model by doubling the kernel size; this leads to significant performance improvement but also slower training time. An interesting direction for further optimization is to first distinguish the blurry and non-blurry regions, e.g. as shown in Fig. 3, and focus more modeling power on blurry regions. This also has applications to object motion modeling in NeRF, as oftentimes only some objects in the scene move when all observations are modeled for deformation. We demonstrate PDRF's application to *object motion modeling* in the supplementary section.

Quantitative and Visual Evaluation. We quantitatively and visually compare PDRF against various radiance field modeling methods in Table. 1 and Fig. 4. NeRF (Mildenhall

Table 1:

	Factory		Cozyroom		Pool		Tanabata		Trolley		Average		Time
Camera Motion	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	Hrs
NeRF	19.32	0.4563	25.66	0.7941	30.45	0.8354	22.22	0.6807	21.25	0.6370	23.78	0.6807	5.00
Plenoxel	18.42	0.4854	25.56	0.8480	30.13	0.8481	21.67	0.6992	20.81	0.6759	23.31	0.7113	0.42
TensoRF	18.85	0.4981	24.91	0.7975	15.89	0.3783	21.06	0.6762	20.01	0.6584	20.53	0.6017	0.83
MPR + NeRF	21.70	0.6153	27.88	0.8502	30.64	0.8385	22.71	0.7199	22.64	0.7141	25.11	0.7476	5.00
Deblur-NeRF	25.60	0.7750	**32.08**	**0.9261**	**31.61**	<u>0.8682</u>	27.11	0.8640	27.45	0.8632	28.77	0.8593	20.0
Deblur-TensoRF	23.20	0.6560	30.18	0.8890	29.83	0.7956	25.13	0.7891	25.10	0.7894	26.69	0.7838	1.50
Deblur-TensoRF[pru]	20.02	0.4889	28.00	0.8570	29.66	0.7968	22.87	0.7029	22.99	0.7231	24.71	0.7137	1.25
PDRF[single]-5	25.65	0.7793	31.37	0.9255	31.28	0.8623	27.55	0.8809	27.43	0.8769	28.66	0.8650	**1.15**
PDRF-5	<u>25.60</u>	<u>0.7786</u>	<u>32.01</u>	0.9310	<u>31.53</u>	**0.8686**	<u>27.70</u>	<u>0.8851</u>	<u>27.90</u>	<u>0.8841</u>	<u>28.95</u>	<u>0.8695</u>	<u>1.33</u>
PDRF-10	**26.56**	**0.8102**	31.90	<u>0.9321</u>	31.29	0.8657	**28.21**	**0.8952**	**28.48**	**0.8956**	**29.29**	**0.8798**	2.33
Defocus	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	Hrs
NeRF	25.36	0.7847	30.03	0.8926	27.77	0.7266	23.80	0.7811	22.67	0.7103	25.93	0.7791	5.00
Plenoxel	25.51	0.8579	30.33	0.9315	27.33	0.7382	23.59	0.8371	22.42	0.7862	25.84	0.8302	0.42
TensoRF	25.08	0.8415	29.77	0.9154	16.23	0.4112	22.97	0.8193	22.40	0.7819	23.17	0.7539	0.83
KPAC + NeRF	26.40	0.8194	28.15	0.8592	26.69	0.6589	24.81	0.8147	23.42	0.7495	25.89	0.7803	5.00
Deblur-NeRF	28.03	0.8628	31.85	0.9175	30.52	0.8246	26.25	0.8517	25.18	0.8067	28.37	0.8527	20.0
Deblur-TensoRF	27.01	0.8257	30.39	0.8882	27.41	0.6841	24.36	0.7835	23.91	0.7593	26.62	0.7882	1.50
Deblur-TensoRF[pru]	25.55	0.7932	28.64	0.8777	27.60	0.6954	23.66	0.7711	22.51	0.7195	25.59	0.7714	1.25
PDRF[single]-5	28.33	0.8743	31.78	0.9222	30.43	0.8233	26.26	0.8649	25.47	0.8261	28.45	0.8622	**1.15**
PDRF-5	<u>30.34</u>	<u>0.9032</u>	<u>32.10</u>	<u>0.9269</u>	<u>30.48</u>	<u>0.8262</u>	<u>27.31</u>	<u>0.8818</u>	<u>27.05</u>	<u>0.8581</u>	<u>29.46</u>	<u>0.8792</u>	1.33
PDRF-10	**30.90**	**0.9138**	**32.29**	**0.9305**	**30.97**	**0.8408**	**28.18**	**0.9006**	**28.07**	**0.8799**	**30.08**	**0.8931**	2.33

Table 1: Quantitative comparisons of different radiance field modeling methods on five synthetic blurry scenes. Best and second best performances are indicated as bold and underlined.

et al. 2020), Plenoxel (Fridovich-Keil et al. 2022), and TensoRF (Chen et al. 2022) are SoTA methods that do not explicitly account for blur degradation. Plenoxel and TensoRF are significantly faster than NeRF; however, these methods often do not outperform NeRF when the inputs are blurry. In fact, TensoRF performs significantly worse and sometimes reaches collapsed solutions, e.g. with the Pool scene, where scene density is all concentrated near the camera origins. This is due to the limited capacity in TensoRF's low-rank density representation, which has more noise given blurry observations. Furthermore, the noise leads to less efficient voxel pruning and longer optimization time. Plenoxel performs better than TensoRF, but takes more memory at 2 GB per scene compared to TensoRF's 200 MB and PDRF's 160 MB features. We note that both Plenoxel and Instant-NGP (Müller et al. 2022) use customized CUDA kernels for acceleration and are difficult to be analyzed or modified. In particular, Instant-NGP does not have baselines for real forward-facing scenes like LLFF (Mildenhall et al. 2019). We also tested on DVGO (Sun et al. 2022); however, DVGO yields collapsed solutions for all scenes.

We then look into radiance field modeling methods that take blur into consideration. One approach is to first apply image-based or video-based deblurring methods on source views, as is done by MPR (Zamir et al. 2021) and KPAC (Son et al. 2021a). While this approach leads to some improvements, denoted as MPR+NeRF and KPAC+NeRF in Table 1, it does not explore 3D redundancy from different source views. Deblur-NeRF (Ma et al. 2022) uses a pixel-coordinate-based blur model before radiance field optimization and does not rely on external training data. In comparison, it outperforms all previous methods; however, Deblur-NeRF is also the most computationally demanding method and requires twenty hours to reach convergence. We create another baseline by replacing NeRF with TensoRF in Deblur-NeRF, which we named Deblur-TensoRF. Deblur-TensoRF is significantly faster than Deblur-NeRF; however, its performance is also notably worse. We contribute this again to TensoRF's compressed density representation. We also note that Deblur-TensoRF does not use voxel pruning. To demonstrate the incompatibility of voxel pruning with dynamic ray generation, we add a Deblur-TensoRF[pru] baseline in Table 1, which prunes low density voxels during training and results in significantly worse performances than Deblur-TensoRF. Our CRR+FVR design uses a combination of explicit and implicit representation to estimate density, and is both faster and more performant than Deblur-TensoRF; specifically, PDRF-5 is 15 times faster than Deblur-NeRF and takes only 80 minutes to converge. This is even faster than Deblur-TensoRF, partially due to the more efficient sampling scheme. We observe that PDRF's PBE can effectively model blur by using
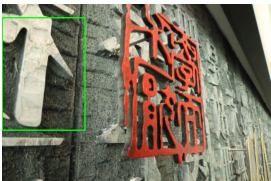
Novel View	Src. View	NeRF 5 hrs	Plenoxel 25 mins	TensoRF 50 mins	Db-Ten.RF 90 mins	Db-NeRF 20 hrs	PDRF 80 mins	G.T.
PSNR/SSIM	Motion	18.98/.538	18.51/.530	16.11/.321	22.00/.726	<u>23.06/.805</u>	**24.66/.870**	
PSNR/SSIM	Defocus	21.71/.798	20.80/.778	19.33/.758	19.33/.758	<u>23.91/.869</u>	**27.92/.928**	
LPIPS	Motion	0.2288	0.2248	0.2722	0.1231	<u>0.0953</u>	**0.0767**	
LPIPS	Defocus	0.1219	0.1582	0.1119	0.0730	<u>0.0669</u>	**0.0599**	

Figure 4: Visual comparisons of different radiance field modeling methods. The first two rows are synthetic scenes with motion and defocus blur. The last two rows are real scenes with motion and defocus blur. For real scenes, the ground truth is for reference as the exposures are inconsistent. Db-Ten.RF and Db-NeRF stands for Deblur-TensoRF and Deblur-NeRF.

feature from the underlying radiance field, especially for defocus blur. This is likely because defocus blur can be much better modeled with 3D context, e.g. scene depth. Camera motion blur is less sensitive to 3D geometry of the scene; however, improvements can still be observed.

Visually, PDRF can recover very fine details both on synthesized and real scenes despite the blurry observations from neighboring source views. As shown in Fig. 4, Deblur-NeRF fails to recover details. While Deblur-TensoRF benefits from the additional blur modeling, it suffers from more noise. This can also be observed when comparing TensoRF to Plenoxel and NeRF. For more visualizations on scene density, please refer to the supplementary material section.

Conclusion

In this work, we present PDRF, a novel radiance field modeling method that addresses camera motion and defocus blur in observations. PDRF is both significantly faster and more performant than previous deblurring methods. Specifically, PDRF proposes PBE, which progressively updates the blur model by incorporating information from the underlying radiance field; as such, PBE can accurately account for the observed blur. To support PBE and accelerate radiance field optimization, PDRF proposes a rendering pipeline that consists of CRR and FVR. CRR is an efficient renderer that predicts ray color based on aggregated ray features. With its lightweight design, CRR provides ray feature estimation to PBE and density estimation to FVR. FVR uses importance sampling to determine efficient voxel samples along the ray, which are then rendered by a larger network for their color and density. Both CRR and FVR use explicit feature representation to help accelerate their convergence rate. We perform extensive experiments and show that our method leads to significantly better deblurring results, especially on defocus blur. PDRF is about 15X faster than Deblur-NeRF, which is the previous SoTA. In the future, we hope to investigate more efficient ways to increase the blur model capacity without sacrificing speed.

References

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR*.

Chakrabarti, A. 2016. A Neural Approach to Blind Motion Deblurring. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907 of *Lecture Notes in Computer Science*, 221–235. Springer.

Chan, T. F.; and Wong, C. 1998. Total variation blind deconvolution. *IEEE Trans. Image Process.*, 7(3): 370–375.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. TensoRF: Tensorial Radiance Fields. In *ECCV*.

Chen, Z.; and Zhang, H. 2019. Learning Implicit Fields for Generative Shape Modeling. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 5939–5948. Computer Vision Foundation / IEEE.

Fridovich-Keil, S.; Yu, A.; Tancik, M.; Chen, Q.; Recht, B.; and Kanazawa, A. 2022. Plenoxels: Radiance Fields Without Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5501–5510.

Han, X.; Laga, H.; and Bennamoun, M. 2021. Image-Based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(5): 1578–1604.

Kim, T. H.; and Lee, K. M. 2015. Generalized video deblurring for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 5426–5434. IEEE Computer Society.

Kim, T. H.; Lee, K. M.; Schölkopf, B.; and Hirsch, M. 2017. Online Video Deblurring via Dynamic Temporal Blending Network. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 4058–4067. IEEE Computer Society.

Krishnan, D.; Tay, T.; and Fergus, R. 2011. Blind deconvolution using a normalized sparsity measure. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, 233–240. IEEE Computer Society.

Kupyn, O.; Budzan, V.; Mykhailych, M.; Mishkin, D.; and Matas, J. 2018. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 8183–8192. Computer Vision Foundation / IEEE Computer Society.

Lee, J.; Son, H.; Rim, J.; Cho, S.; and Lee, S. 2021. Iterative Filter Adaptive Network for Single Image Defocus Deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 2034–2042. Computer Vision Foundation / IEEE.

Levin, A. 2006. Blind Motion Deblurring Using Image Statistics. In Schölkopf, B.; Platt, J. C.; and Hofmann, T., eds., *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, 841–848. MIT Press.

Li, D.; Xu, C.; Zhang, K.; Yu, X.; Zhong, Y.; Ren, W.; Suominen, H.; and Li, H. 2021. ARVo: Learning All-Range Volumetric Correspondence for Video Deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 7721–7731. Computer Vision Foundation / IEEE.

Ma, L.; Li, X.; Liao, J.; Zhang, Q.; Wang, X.; Wang, J.; and Sander, P. V. 2022. Deblur-NeRF: Neural Radiance Fields from Blurry Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12861–12870.

Martin-Brualla, R.; Radwan, N.; Sajjadi, M. S. M.; Barron, J. T.; Dosovitskiy, A.; and Duckworth, D. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 7210–7219. Computer Vision Foundation / IEEE.

Mescheder, L. M.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; and Geiger, A. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 4460–4470. Computer Vision Foundation / IEEE.

Mildenhall, B.; Hedman, P.; Martin-Brualla, R.; Srinivasan, P. P.; and Barron, J. T. 2022. NeRF in the Dark: High Dynamic Range View Synthesis From Noisy Raw Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16190–16199.

Mildenhall, B.; Srinivasan, P. P.; Cayon, R. O.; Kalantari, N. K.; Ramamoorthi, R.; Ng, R.; and Kar, A. 2019. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4): 29:1–29:14.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, 405–421. Springer.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.*, 41(4): 102:1–102:15.

Nah, S.; Kim, T. H.; and Lee, K. M. 2017. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 257–265. IEEE Computer Society.

Nah, S.; Son, S.; and Lee, K. M. 2019. Recurrent Neural Networks With Intra-Frame Iterations for Video Deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 8102–8111. Computer Vision Foundation / IEEE.

Pan, J.; Bai, H.; and Tang, J. 2020. Cascaded Deep Video Deblurring Using Temporal Sharpness Prior. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 3040–3048. Computer Vision Foundation / IEEE.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R. A.; and Lovegrove, S. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 165–174. Computer Vision Foundation / IEEE.

Park, K.; Sinha, U.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Seitz, S. M.; and Martin-Brualla, R. 2021a. Nerfies: Deformable Neural Radiance Fields. *ICCV*.

Park, K.; Sinha, U.; Hedman, P.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Martin-Brualla, R.; and Seitz, S. M. 2021b. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph.*, 40(6).

Pumarola, A.; Corona, E.; Pons-Moll, G.; and Moreno-Noguer, F. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 10318–10327. Computer Vision Foundation / IEEE.

Rudin, L. I.; Osher, S.; and Fatemi, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1): 259–268.

Sitzmann, V.; Zollhöfer, M.; and Wetzstein, G. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 1119–1130.

Son, H.; Lee, J.; Cho, S.; and Lee, S. 2021a. Single Image Defocus Deblurring Using Kernel-Sharing Parallel Atrous Convolutions. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2622–2630.

Son, H.; Lee, J.; Lee, J.; Cho, S.; and Lee, S. 2021b. Recurrent Video Deblurring with Blur-Invariant Motion Estimation and Pixel Volumes. *ACM Trans. Graph.*, 40(5): 185:1–185:18.

Sun, C.; et al. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Tai, Y.-W.; et al. 2011. Richardson-Lucy Deblurring for Scenes under a Projective Motion Path. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8): 1603–1618.

Tancik, M.; Srinivasan, P. P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J. T.; and Ng, R. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS*.

Tao, X.; Gao, H.; Shen, X.; Wang, J.; and Jia, J. 2018. Scale-Recurrent Network for Deep Image Deblurring. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 8174–8182. Computer Vision Foundation / IEEE Computer Society.

Tewari, A.; Fried, O.; Thies, J.; Sitzmann, V.; Lombardi, S.; Sunkavalli, K.; Martin-Brualla, R.; Simon, T.; Saragih, J. M.; Nießner, M.; Pandey, R.; Fanello, S. R.; Wetzstein, G.; Zhu, J.; Theobalt, C.; Agrawala, M.; Shechtman, E.; Goldman, D. B.; and Zollhöfer, M. 2020. State of the Art on Neural Rendering. *Comput. Graph. Forum*, 39(2): 701–727.

Tretschk, E.; Tewari, A.; Golyanik, V.; Zollhöfer, M.; Lassner, C.; and Theobalt, C. 2021. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Xu, L.; Zheng, S.; and Jia, J. 2013. Unnatural L0 Sparse Representation for Natural Image Deblurring. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 1107–1114. IEEE Computer Society.

Zamir, S. W.; Arora, A.; Khan, S. H.; Hayat, M.; Khan, F. S.; Yang, M.; and Shao, L. 2021. Multi-Stage Progressive Image Restoration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 14821–14831. Computer Vision Foundation / IEEE.

Zhang, K.; Luo, W.; Zhong, Y.; Ma, L.; Stenger, B.; Liu, W.; and Li, H. 2020. Deblurring by Realistic Blurring. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2734–2743. Computer Vision Foundation / IEEE.

Zhong, Z.; Gao, Y.; Zheng, Y.; Zheng, B.; and Sato, I. 2021. Efficient Spatio-Temporal Recurrent Neural Network for Video Deblurring. *CoRR*, abs/2106.16028.