# Deblur-NeRF: Neural Radiance Fields from Blurry Images

Li Ma[1*] Xiaoyu Li[2] Jing Liao[3] Qi Zhang[2] Xuan Wang[2]
Jue Wang[2] Pedro V. Sander[1]

[1]The Hong Kong University of Science and Technology
[2]Tencent AI Lab [3]City University of Hong Kong


(a) Samples of Blurry Source Views (b) Novel Views from NeRF (c) Novel Views from Deblur-NeRF

Figure 1. Given a set of blurry multi-view input images (a), the original NeRF implementation reconstructs blurry novel views (b). Our method is able to recover a sharp radiance field and synthesize clear novel views (c). Our proposed approach can handle both camera motion blur (first row) and defocus blur (second row). Please refer to the *supplementary material* for video results.

## Abstract

*Neural Radiance Field (NeRF) has gained considerable attention recently for 3D scene reconstruction and novel view synthesis due to its remarkable synthesis quality. However, image blurriness caused by defocus or motion, which often occurs when capturing scenes in the wild, significantly degrades its reconstruction quality. To address this problem, We propose Deblur-NeRF, the first method that can recover a sharp NeRF from blurry input. We adopt an analysis-by-synthesis approach that reconstructs blurry views by simulating the blurring process, thus making NeRF robust to blurry inputs. The core of this simulation is a novel Deformable Sparse Kernel (DSK) module that models spatially-varying blur kernels by deforming a canonical sparse kernel at each spatial location. The ray origin of each kernel point is jointly optimized, inspired by the physical blurring process. This module is parameterized as an MLP that has the ability to be generalized to various blur types. Jointly optimizing the NeRF and the DSK module allows us to restore a sharp NeRF. We demonstrate that our method can be used on both camera motion blur and defocus blur: the two most common types of blur in real scenes. Evaluation results on both synthetic and real-world data show that our method outperforms several baselines. The synthetic and real datasets along with the source code is publicly available at* https://limacv.github.io/deblurnerf/.

## 1. Introduction

Tremendous progress has been witnessed in the past few years in novel view synthesis, where an intermediate 3D representation is reconstructed from sparse input views to interpolate or extrapolate arbitrary novel views. Recently, NeRF [22] emerged as an effective scene representation that achieves photorealistic rendering results. It models a static scene as a continuous volumetric function that maps 3D location and 2D view direction to color and density. This function is parameterized as a multilayer perceptron (MLP), and its output can be rendered by volume rendering techniques in a differentiable manner.

To reconstruct a NeRF, several images from different views are needed. While the original approach for train-

*Author did this work during the internship at Tencent AI Lab.

ing NeRF works well when these images are well captured and calibrated, it would produce obvious artifacts when blur occurs. For example, when using a long exposure setting to capture a low-light scene, the images are more sensitive to camera shake, resulting in camera motion blur. Furthermore, defocus blur is inevitable when capturing scenes with large depth variation using a large aperture. These blurs will significantly decrease the quality of the reconstructed NeRF, resulting in artifacts in the rendered novel views.

Many works have recently been proposed to tackle abnormal input while training NeRF. NeRF-W [20] focuses on images with illumination change and moving objects. Mip-NeRF [1] improves the NeRF when the input spans different scales. Distortion in input is considered and calibrated simultaneously in SCNeRF [11]. To the best of our knowledge, none has considered addressing the problem of training NeRF from blurry input images. One solution is to first deblur the input in image space and then train the NeRF with deblurred images, which we refer to as the image-space baseline. This baseline improves the novel view synthesis quality of NeRF to some extend by utilizing recent single image or video deblurring methods. However, the single-image deblur methods fail to aggregate information from neighbor views and can not guarantee the multi-view consistent result. Video-based methods manage to take multi-frame into consideration, usually relying on image space operations such as optical flows and feature correlation volumes. However, these methods fail to exploit the 3D geometry of the scene, leading to inaccurate correspondences across views, especially when they have a large baseline. On the contrary, our method deblurs by aggregating information from all observations with full awareness of the 3D scene.

In this paper, we propose Deblur-NeRF, an effective framework that explicitly models the blurring process in the network, and is capable of restoring a sharp NeRF from blurry input. We model the blurring process by convolving a clean image using a blur kernel similar to blind deconvolution methods [2]. A novel deformable sparse kernel (DSK) module is proposed to model the blur kernel inspired by the following observations. First, convolving with dense kernels is infeasible for scene representations such as NeRF due to the dramatic increase in computation and memory usage during rendering. To address this, DSK uses sparse rays to approximate the dense kernel. Second, we show that the actual blurring process involves combining rays from different origins, which motivates us to jointly optimize the ray origins. Finally, to model spatially-varying blur kernels, we deform a canonical sparse kernel at each 2D spatial location. The deformation is parameterized as an MLP that can be generalized to different types of blur. During training, we jointly optimize the DSK and a sharp NeRF with only blurry input as supervision, while in the inference stage, clear novel views can be rendered by removing the DSK. We conduct extensive experiments on both synthetic and real datasets with two types of blur: camera motion blur and defocus blur. Results show that the proposed method outperforms the original NeRF and image-space baselines (i.e., combining NeRF with the state-of-the-art image or video deblurring methods), for these two blur types, as shown in Fig. 1 and the experiments section. Our contributions can be summarized as follows:

- We propose the first framework that can reconstruct a sharp NeRF from blurry input.
- We propose a deformable sparse kernel module that enables us to effectively model the blurring process and is generalizable for different types of blur.
- We analyze the physical blurring process and extend the 2D kernel to 3D space by considering the translation of the ray origin for each kernel point.

## 2. Related Work

**Neural radiance field.** Our work extends the NeRF [22], a coordinate-based implicit 3D scene representation, which has gained popularity over the past few years due to its state-of-the-art novel view synthesis results. The success of NeRF has inspired many follow-up works that extend the NeRF [7, 8, 18, 25, 30–32, 34, 38]. Several works have explored to train the NeRF with non-ideal input. For example, BRAF [19], NeRF−− [47] and GNeRF [21] try to train the NeRF without camera poses. SCNeRF [11] focuses on jointly calibrating a more complex non-linear camera model. To address the NeRF training under uncontrolled, in-the-wild photographs, NeRF-W [20] introduces several extensions to NeRF that successfully model the inconsistent appearance variations and transient objects across views. PixelNeRF [51] reconstructs a neural volume with only one or few images. Moreover, Jonathan *et al*. proposes Mip-NeRF [1] which improves the NeRF under input with different scales, producing anti-aliased results. However, training NeRF with blurry images is still an unexplored area, as none of the aforementioned works seem to explicitly consider this kind of degradation.

**Single image deblurring.** Image deblurring aims at recovering a sharp image from blurry input. Usually, the blurry image is modeled as the sharp image convolved with a blur kernel, and the deblurring process is formulated as jointly solving the sharp image and the kernel given the blurry observation. This task is ill-posed since there are many sets of image-blur pairs that can synthesize the observed blurry image [46]. Classical blind deblur algorithms tackle the ill-posedness by introducing hand-crafted or learned image priors while optimizing the sharp image and kernel, such as total variation [4, 35], normalized gradient sparsity [13] and unnatural $l_0$ [49]. Since blur in

real world photographs is usually spatially-varying, many works try to reparameterize the blur kernels to a smaller solve space. Early work uses projective motion blur [42] which fits spatially-varying blur kernels using multiple homography, while region-based methods assume piece-wise constant [16] or piece-wise projective [28]. Moreover, a depth-based model is used to optimize the depth map and camera poses jointly [29, 33]. Another approach to model blur kernels is to use optical flow [9]. These methods either make a strong assumption on the blur pattern, or can only model one specific type of blur. In contrast, our method models the spatially-varying kernel using MLP, which can be generalized to different blur types. The recent trend of image deblurring is to introduce deep neural networks that directly map the blurry image to the latent sharp image [3, 14, 15, 23, 26, 39, 44, 48, 52, 53]. These approaches have outperformed traditional methods. However, this line of work highly depends on the training data and the methods often have difficulty generalizing to unseen blur types in the real world [45].

**Multi-image deblurring.** Deblurring with multi-image settings poses new challenges in aggregating information across frames and preserving temporal consistency. Optical flow is a useful tool for registering neighbor frames to the reference frame [9, 27]. However, estimating accurate optical flow is difficult and ill-posed, especially when the input is blurry. With the advancement of deep learning, one can design flow-free methods by concatenating multiple frames and directly restoring the clean frame using a CNN [41]. Another option is to use recurrent structure that propagates features across frames [10, 24, 40, 56]. Li *et al*. [17] extends the optical flow to feature correlation volume, which greatly improves the performance. Similarly, Son *et al*. [40] propose pixel volume that relaxes the requirement for accurate flow. However, these multi-image deblurring methods, which are built on image space operations, fail to exploit the 3D geometry of the scene and have difficulty addressing the multi-view input with a large baseline.

## 3. Preliminary

We first review the NeRF representation [22] for a static 3D scene. A NeRF defines the scene as a continuous volumetric function that maps a 3D position $\mathbf{x}$ and a 2D view direction $\mathbf{d}$ to color $\mathbf{c}$ and volume density $\sigma$. Formally:

$$(\mathbf{c}, \sigma) = F_{\mathbf{\Theta}}\left(\gamma_{L_x}(\mathbf{x}), \gamma_{L_d}(\mathbf{d})\right), \tag{1}$$

where $F_{\mathbf{\Theta}}$ represents an MLP with parameters $\mathbf{\Theta}$, and $\gamma_L(\cdot)$ is the positional encoding that maps each element of a vector into a higher dimensional frequency space:

$$\gamma_L(x) = \left[\sin \pi x, \cos \pi x, ..., \sin 2^{L-1}\pi x, \cos 2^{L-1}\pi x\right]^{\mathrm{T}}, \tag{2}$$

where the hyper-parameter $L$ indicates the highest frequency used in the mapping and can be used to control the smoothness of the scene function [43]. To render a pixel centered at image coordinate $\mathbf{p}$, we first emit a ray $\mathbf{r_p}(t) = \mathbf{o} + t\mathbf{d_p}$ from camera projection center $\mathbf{o}$ along the viewing direction $\mathbf{d_p}$. Then a sampling strategy is used to determine $D$ sorted distances $\{t^{(i)}\}_{i=1}^{D}$ between predefined near and far planes $t^{(0)}$ and $t^{(D+1)}$. We estimate the color $\mathbf{c}^{(i)}$ and density $\sigma^{(i)}$ at each sample point $\mathbf{r_p}(t^{(i)})$ using $F_{\mathbf{\Theta}}$. The final color of the pixel is computed as:

$$\hat{\mathbf{c}}_{\mathbf{p}} = \hat{\mathbf{c}}(\mathbf{r_p}) = \sum_{i=1}^{D} T^{(i)} \left(1 - \exp(-\sigma^{(i)}\delta^{(i)})\right) \mathbf{c}^{(i)}, \tag{3}$$

where $\delta^{(i)} = t^{(i+1)} - t^{(i)}$ is the distance between adjacent samples, and $T^{(i)} = \exp(-\sum_{j=1}^{i-1} \sigma^{(j)}\delta^{(j)})$. In this paper we use $\mathbf{c_p}$ and $\mathbf{c}(\mathbf{r_p})$ interchangeably. Note that this rendering process is trivially differentiable.

## 4. Method

Our task is to train the NeRF with blurry input. The training pipeline is visualized in Fig. 2. The core idea is to explicitly model the blurring process and seek to optimize a sharp NeRF and the blur parameters jointly so that the synthesized blurry images match the input. Specifically, to render a blurry pixel during training, we first generate multiple optimized rays using a newly proposed Deformable Sparse Kernel (DSK) module, which mimics the blurring process. We render these rays using the NeRF, and blend results to get the finally blurry color, which is then supervised by the blurry input. Note that in the inference stage, we can directly render the NeRF without the DSK to get sharp novel views. We describe the DSK as well as some other designs in the following subsections.

### 4.1. Deformable Sparse Kernel

Akin to most image deblurring algorithms [2, 46], we model the blurring process by convolving the sharp image with a blur kernel $h$:

$$\mathbf{b_p} = \mathbf{c_p} * h, \tag{4}$$

where $\mathbf{c_p}$ is the color of sharp pixel at $\mathbf{p}$, which ideally is also the output of a sharp NeRF in our model. $\mathbf{b_p}$ is the corresponding blurry color and $*$ stands for the convolution operator. The support of the blur kernel is usually defined in a $K \times K$ window centered at $\mathbf{p}$. To compute $\mathbf{b_p}$, we take the sum of element-wise multiplication of $\mathbf{c_p}$ and $h$ inside the window. This can be computed efficiently in the map-based image representation. However, a problem arises when $\mathbf{c_p}$ is modeled as a NeRF because rendering becomes quite computation and memory consuming. For each pixel, there are $K \times K$ rays that need to be rendered

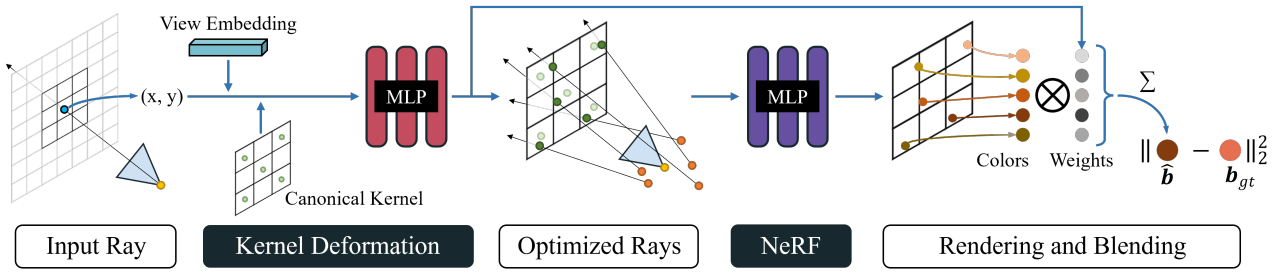



Figure 2. An overview of our training framework. When rendering a ray, we first predict $N$ sparse optimized rays based on a canonical kernel along with their weights. After rendering these rays, we combine the results to get the blurry pixel $\hat{\mathbf{b}}$. Note that when testing, we can directly render the rays without kernel deformation resulting in a sharp image.

in the supporting window, thus making training infeasible. Therefore, we propose to approximate the dense blur kernel with a small number of sparse points:

$$\mathbf{b_p} = \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} w_\mathbf{q}\mathbf{c_q}, \text{ w.r.t. } \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} w_\mathbf{q} = 1, \tag{5}$$

where $\mathcal{N}(\mathbf{p})$ is the set of $N$ locations sparsely distributed around $\mathbf{p}$ that compose the support of our sparse kernel. $w_\mathbf{q}$ is the corresponding weight at each location. We set $N$ to be a fixed number and ablate this hyper-parameter in our experiments (Sec. 5). Note that $\mathbf{q}$ is a continuous value and we can jointly optimize the locations $\mathcal{N}(\mathbf{p})$, $w_\mathbf{q}$ and the NeRF so that the best sparse kernel is regressed.

The blur kernel is usually spatially-varying in real world images. Inspired by the NeRF that uses an MLP as a continuous 5D function, we also choose to use an MLP to model the spatially-varying kernel. Specifically, for each input view, we assign "canonical kernel locations" $\mathcal{N}'(\mathbf{p}) = \{\mathbf{q}'_i\}_{i=0}^{N-1}$, and use an MLP to deform the locations and while also predicting the weights:

$$(\Delta\mathbf{q}, w_\mathbf{q}) = G_\mathbf{\Phi}(\mathbf{p}, \mathbf{q}', \mathbf{l}), \text{where } \mathbf{q}' \in \mathcal{N}'(\mathbf{p}). \tag{6}$$

Here $G_\mathbf{\Phi}$ indicates an MLP with parameter $\mathbf{\Phi}$ and $\mathbf{l}$ is a learned view embedding. This view embedding is necessary since the blur patterns usually differ across views. Optimizing a different view embedding for each view allows the DSK module to fit a different blur kernel for each view. In our experiments we set the view embedding to be a vector of length 32. We compute the final sparse kernel location in $\mathcal{N}(\mathbf{p})$ as $\mathbf{q} = \mathbf{q}' + \Delta\mathbf{q}$. Note that we need to forward the MLP $G_\mathbf{\Phi}$ for $N$ times to get all the deformed locations. One option that may potentially boost the performance is introducing positional encoding to $G_\mathbf{\Phi}$ by replacing the input $\mathbf{p}$ in Eq. (6) with $\gamma(\mathbf{p})$. However, we find this operation does not help to improve the quality. One possible reason is that the spatially-varying kernel changes gradually along spatial positions without high-frequency variation.

### 4.2. Convolution with Irradiance

As pointed out by Chen *et al.* [5], this blur convolution model should be applied to scene irradiance instead of image intensity. A more physically correct model should be $\mathbf{b}_p = f(\mathbf{c}'_p * h)$, where $\mathbf{c}'$ indicates the scene irradiance and $f(\cdot)$ is the camera response function (CRF) that maps the scene irradiance to image intensity. A nonlinear CRF will increase the complexity of the blur kernel and make the learning of DSK difficult if the linear model in Eq. (4) is used, especially in high contrast regions [46]. To compensate for the nonlinear CRF, we assume that our sharp NeRF predicts colors in linear space and adopt a simple gamma correction function in the final output:

$$\mathbf{b_p} = g(\sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} w_\mathbf{q}\mathbf{c}'_\mathbf{q}), \tag{7}$$

where $g(\mathbf{c}') = \mathbf{c}'^{\frac{1}{2.2}}$ is the gamma correction function. More complex CRFs could be used to model real world cameras, such as pre-calibrated CRFs, or jointly optimizing the CRFs during training. But we find this simple scheme is enough to compensate the nonlinearity in the imaging process and improve the quality. More discussions about modeling the CRF can be found in *supplementary material*.

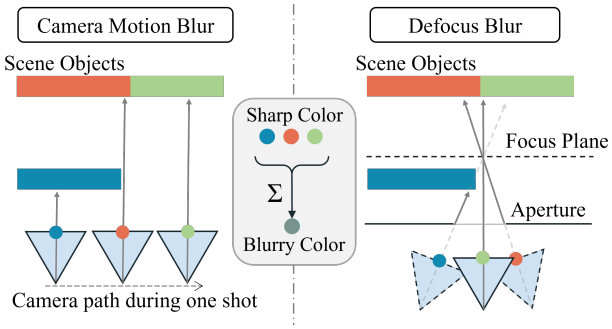



Figure 3. Top-down view of the blurring processes caused by camera motion and defocus. The blurry color is blended from all the rays destined at a pixel. In defocus blur, the rays of a pixel are scattered to different directions at the focus plane, which is equivalent to a mixture of rays being emitted from different camera centers.

### 4.3. Optimizing the Ray Origin

One observation of the convolutional model is that it is a 2D approximation of the actual blur model. In the convo-

lutional model, the blurry result is a combination of neighbor pixels, which are the rendering results of neighbor rays with the same camera center as the origin. However, the actual blurring process usually involves blending rays cast from different origins. Consider the two blurring processes shown in Fig. 3. When capturing camera motion blur, the camera center moves during one shot, leading to the change of the ray origins. And for defocus blur, the ray gets scattered to different directions which is equivalent to a mixture of rays from different origins. When the scene is mostly planar, the translation of the ray origin can be well approximated by the 2D translation of the pixel position. However, due to the parallax effect and occlusion, this is not the case when there is depth discontinuity. Since we have access to the 3D scene representation, we can develop kernels that consider the change of ray origins. Thus we jointly optimize the translation of the ray origin of each sparse kernel location. Specifically, we jointly predict the origin translation for each kernel location as follows Eq. (6):

$$(\Delta\mathbf{o_q}, \Delta\mathbf{q}, w_\mathbf{q}) = G_\mathbf{\Phi}(\mathbf{p}, \mathbf{q}', \mathbf{l}), \mathbf{q}' \in \mathcal{N}'(\mathbf{p}), \quad (8)$$

and then generate rays by:

$$\mathbf{r_q} = (\mathbf{o} + \Delta\mathbf{o_q}) + t\mathbf{d_q}, \mathbf{q} = \mathbf{q}' + \Delta\mathbf{q}. \quad (9)$$

These optimized rays are rendered and combined to get the final blurry pixels.

The training process is summarized as follows: we first predict tuples $\{\Delta\mathbf{o_q}, \Delta\mathbf{q}, w_\mathbf{q}\}_{q\in\mathcal{N}(p)}$ using Eq. (8), with which we generate multiple optimized rays $\{\mathbf{r_q}\}_{q\in\mathcal{N}(p)}$ by deforming the canonical sampling location and optimizing the ray origin as in Eq. (9). We render these rays to get $\mathbf{c}'_q$ using Eq. (3) and blend to get a blurry pixel $\hat{\mathbf{b}}_p$ using Eq. (7). This synthetic blurry pixel is supervised by the corresponding ground truth pixel color $\mathbf{b}_{gt}$:

$$\mathcal{L}_{reconstruct} = \sum_{\mathbf{p}\in\mathcal{R}} \|\hat{\mathbf{b}}_p - \mathbf{b}_{gt}\|_2^2, \quad (10)$$

where $\mathcal{R}$ is the set of pixels in each batch. Note that our pipeline is only used during training. At test time, we can directly render the sharp results using the restored sharp NeRF with gamma correction.

### 4.4. Aligning the NeRF

As shown in the experiments, if we freely optimize all the learnable components, i.e., the NeRF and the deformable sparse kernel, the reconstructed NeRF may undergo some non-rigid distortion. This is in line with expectations because it is possible that the scene represented by the NeRF and the learned kernel deform together without affecting the reconstructed blurry result. However this is usually not desired. To constrain the NeRF model to align with the observations, we first initialize the deformable sparse kernel so that all the optimized rays $\mathbf{r_q}$ are close to the input ray $\mathbf{r_p}$. This is implemented by multiplying a small gain of $\epsilon = 0.1$ to each element of the output tuples $(\Delta\mathbf{o_q}, \Delta\mathbf{q}, w_\mathbf{q})$. As a result, the ray origins $\mathbf{o_q}$ and the kernel points $\mathbf{q}$ are initialized to be close to the camera centers and the pixel locations, respectively. And all kernel points will have roughly the same weights at the beginning of training. We additionally introduce an alignment loss that forces one of the optimized rays $\mathbf{r_q}$ to be similar to the input ray $\mathbf{r_p}$:

$$\mathcal{L}_{align} = \|\mathbf{q}_0 - \mathbf{p}\|_2 + \lambda_o\|\Delta\mathbf{o}_{\mathbf{q}_0}\|_2, \quad (11)$$

where $\mathbf{q}_0$ is a fixed element in $\mathcal{N}(\mathbf{q})$. By applying $\mathcal{L}_{align}$, we supervise $\mathbf{q}_0$ to be the center of the kernel. We set $\lambda_o = 10$ in all of our experiments.

Our final loss is a combination of the NeRF reconstruction loss and the alignment loss:

$$\mathcal{L} = \mathcal{L}_{reconstruct} + \lambda_a\mathcal{L}_{align}. \quad (12)$$

In our experiments we set $\lambda_a = 0.1$.

## 5. Experiments

### 5.1. Implementation Details

**Training.** We build our deformable sparse kernel on the Pytorch re-implementation of the NeRF [50]. We use a batch size of 1024 rays, each sample at 64 coordinates in the coarse volume and 64 additional coordinates in the fine volume. We set the number of sparse locations $N = 5$. We use the Adam optimizer [12] with default parameters. We schedule the learning rate to start at $5 \times 10^{-4}$ and decay exponentially to $8 \times 10^{-5}$ over the coarse of the optimization. We train each scene for 200k iterations on a single NVIDIA V100 GPU. We adopt the same MLP structure of $F_\mathbf{\Theta}$ as the original NeRF [22], and for $G_\mathbf{\Phi}$, we use MLP with 4 fully-connected hidden layers, each layer having 64 channels and ReLU activations. We also add a shortcut that connects the first layer to the last layer.

**Datasets.** In the experiments we focus on two blur types: camera motion blur and defocus blur. For each type of blur we synthesize 5 scenes using Blender [6]. We manually place multi-view cameras to mimic real data capture. To render images with camera motion blur, we randomly perturb the camera pose, and then linearly interpolate poses between the original and perturbed poses for each view. We render images from interpolated poses and blend them in linear RGB space to generate the final blurry images. For defocus blur, we use the built-in functionality to render depth-of-field images. We fix the aperture and randomly choose a focus plane between the nearest and furthest depth.

We also captured 20 real world scenes with 10 scenes for each blur type for a qualitative study. The camera used

| *Camera Motion* | FACTORY | | | COZYROOM | | | POOL | | | TANABATA | | | TROLLEY | | | AVERAGE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| w/o gamma | 23.27 | .6908 | .3210 | 29.86 | .8964 | .0564 | 31.29 | .8635 | .1339 | 25.50 | .8211 | .1472 | 25.44 | .8071 | .1513 | 27.07 | .8158 | .1620 |
| w/o align | 24.95 | .7419 | .2780 | 25.53 | .8032 | .0636 | 27.45 | .7389 | .1443 | 24.77 | .8086 | .1282 | 24.71 | .8036 | .1324 | 25.48 | .7792 | .1493 |
| w/o origin opt. | 25.29 | .7657 | .2827 | 31.86 | .9244 | .0479 | 31.64 | .8691 | .1216 | 26.20 | .8475 | .1523 | 25.53 | .8199 | .1774 | 28.11 | .8453 | .1564 |
| Ours | 25.60 | .7750 | .2687 | 32.08 | .9261 | .0447 | 31.61 | .8682 | .1246 | 27.11 | .8640 | .1228 | 27.45 | .8632 | .1363 | 28.77 | .8593 | .1400 |
| *Defocus* | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| w/o gamma | 25.92 | .8170 | .1579 | 31.18 | .9078 | .0556 | 29.97 | .8095 | .2082 | 25.00 | .8172 | .1299 | 24.37 | .7819 | .1680 | 27.29 | .8267 | .1439 |
| w/o align | 26.31 | .8051 | .1493 | 27.76 | .8583 | .0545 | 28.00 | .7481 | .2078 | 24.86 | .8149 | .1021 | 24.50 | .7977 | .1292 | 26.28 | .8048 | .1286 |
| w/o origin opt. | 28.00 | .8584 | .1344 | 31.85 | .9173 | .0506 | 30.21 | .8172 | .2023 | 25.75 | .8444 | .1104 | 24.82 | .8014 | .1590 | 28.13 | .8477 | .1313 |
| Ours | 28.03 | .8628 | .1127 | 31.85 | .9175 | .0481 | 30.52 | .8246 | .1901 | 26.25 | .8517 | .0995 | 25.18 | .8067 | .1436 | 28.37 | .8527 | .1188 |

Table 1. Ablations of our method in the synthetic scenes. We separately report numeric results of two blur types: camera motion blur and defocus blur. We color code each row as **best** and **second best**.


(a) w/o gamma (b) w/o align (c) w/o origin opt. (d) Ours (e) Ground Truth

Figure 4. Examples of ablations in the synthetic scenes. The corresponding error map is visualized in the bottom, where darker regions indicate smaller error. Our full model has the smallest error, especially at the edges. Note the artifacts highlighted in the red boxes.

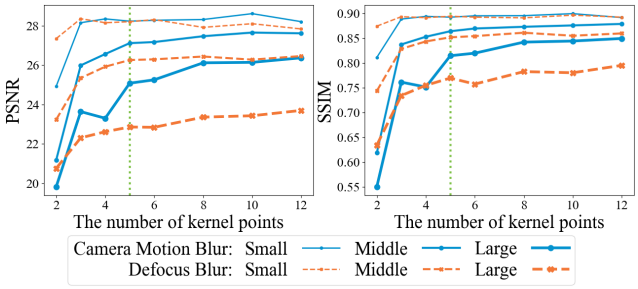



Figure 5. Comparison of our full model using a different number of kernel points. The vertical green lines indicate the $N = 5$, which we use in our other experiments.

was a Canon EOS RP with manual exposure mode. We captured the camera motion blur images by manually shaking the camera during exposure, while the reference images are taken using a tripod. To capture defocus images, we choose a large aperture. We compute the camera poses of blurry and reference images in the real world scenes using the COLMAP [36, 37]. Although the estimated poses from COLMAP may be ambiguous due to the blur, we find our method is robust to inaccurate poses. One reason is that optimizing ray origins compensate for the registration errors.

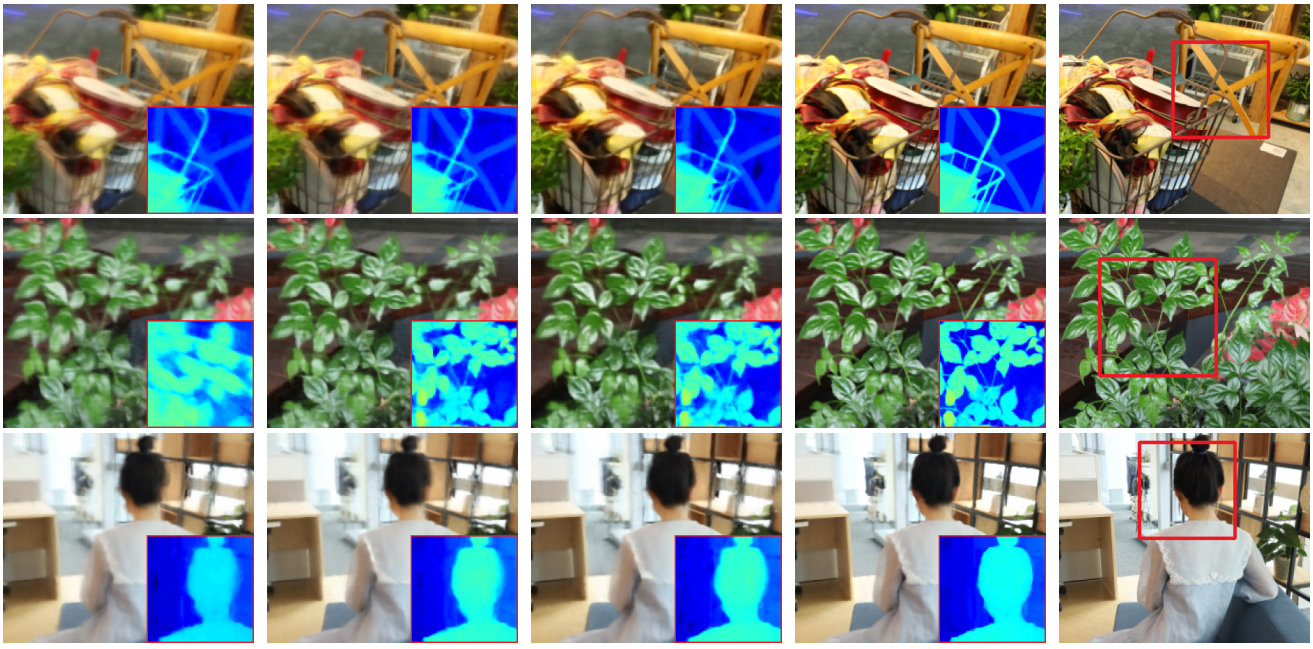
## 5.2. Ablation Study

**Effectiveness of main components.** We first conduct ablations on several components in our framework: gamma correction (w/o gamma), ray origin optimization (w/o origin opt.), and the alignment loss (w/o align). We remove these components individually and train a separate NeRF in each of the synthetic camera motion blur and defocus blur datasets. We report the PSNR, SSIM, and LPIPS [55] metrics between the synthesized novel views and ground truth novel views. As shown in Tab. 1, overall, the best result is achieved when the full model is used. We visualize the results and the error maps of two examples in the Fig. 4. We note that other methods present larger errors, especially at the boundaries of objects. Without $\mathcal{L}_{align}$, the NeRF produces sharp but misaligned novel views.

**Number of kernel points.** One important hyper-parameter in our method is the number of sparse locations $N = |\mathcal{N}(\mathbf{p})|$. We experimented with different values of $N$ with our full model in various blur situations, considering both blur types and three degrees of blur. We plot the PSNR and SSIM curves in Fig. 5. We note that in all cases the quality of the results improves as $N$ increases. However, the improvement is less substantial beyond $N = 5$. In case

| *Camera Motion* | FACTORY | | | COZYROOM | | | POOL | | | TANABATA | | | TROLLEY | | | AVERAGE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| *naive NeRF* | 19.32 | .4563 | .5304 | 25.66 | .7941 | .2288 | 30.45 | .8354 | .1932 | 22.22 | .6807 | .3653 | 21.25 | .6370 | .3633 | 23.78 | .6807 | .3362 |
| *MPR + NeRF* | 21.70 | .6153 | .3094 | 27.88 | .8502 | .1153 | 30.64 | .8385 | .1641 | 22.71 | .7199 | .2509 | 22.64 | .7141 | .2344 | 25.11 | .7476 | .2148 |
| *PVD + NeRF* | 20.33 | .5386 | .3667 | 27.74 | .8296 | .1451 | 27.56 | .7626 | .2148 | 23.44 | .7293 | .2542 | 23.81 | .7351 | .2567 | 24.58 | .7190 | .2475 |
| Ours | 25.60 | .7750 | .2687 | 32.08 | .9261 | .0477 | 31.61 | .8682 | .1246 | 27.11 | .8640 | .1228 | 27.45 | .8632 | .1363 | 28.77 | .8593 | .1400 |
| *Defocus* | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| *naive NeRF* | 25.36 | .7847 | .2351 | 30.03 | .8926 | .0885 | 27.77 | .7266 | .3340 | 23.80 | .7811 | .2142 | 22.67 | .7103 | .2799 | 25.93 | .7791 | .2303 |
| *KPAC + NeRF* | 26.40 | .8194 | .1624 | 28.15 | .8592 | .0815 | 26.69 | .6589 | .2631 | 24.81 | .8147 | .1639 | 23.42 | .7495 | .2155 | 25.89 | .7803 | .1773 |
| Ours | 28.03 | .8628 | .1127 | 31.85 | .9175 | .0481 | 30.52 | .8246 | .1901 | 26.25 | .8517 | .0995 | 25.18 | .8067 | .1436 | 28.37 | .8527 | .1188 |

Table 2. Quantitative comparison on synthetic scenes of two blur types. We color code each row as **best** and **second best**



*naive NeRF* *MPR + NeRF* *PVD + NeRF* Ours Reference

Figure 6. Qualitative comparison on real world camera motion blur. The last column is captured for reference only and may be misaligned with the ground truth.

the input is extremely blurry, further increasing the number of kernel points can potentially help. However, increasing $N$ comes at a larger computation and memory cost during training. Therefore, we use $N = 5$ for all other experiments, providing a good balance between rendering quality and efficiency.

### 5.3. Comparisons

Since there are no existing works that try to reconstruct the NeRF from blurry input for novel view synthesis, we carefully select several possible baselines to compare with. The most straightforward one would be to directly train the NeRF using the blurry input (*naive NeRF*). Additionally, we also compare to the image-space baselines that first restore the input using the existing image or video based deblurring techniques and then train the NeRF with the deblurred images. For camera motion blur, we compare with the current state-of-the-art methods for deblurring on single image [52] (*MPR + NeRF*) and video [40] (*PVD + NeRF*). For defocus blur, we compare to the KPAC [39] (*KPAC + NeRF*). We show quantitative results on synthetic scenes in Tab. 2. For real scenes, due to the nature of capturing, the ground truth images are not available due to the misalignment for camera motion blur or exposure variations for defocus blur. We can see that while the image-space deblurring baseline improves compared to the *naive NeRF* baseline, our full pipeline outperforms these baselines to a large extent in the synthetic scenes among two blur types. The video based deblurring method *PVD + NeRF* sometimes performs worse than the single-image based method *MPR + NeRF*. A possible reason is that *PVD + NeRF* aggregates features from neighbor frames based on optical flow, which is really challenging for blurry input with large baselines. Fig. 6 and 7 showcase the qualitative comparison results on camera motion blur and defocus blur, respectively, in real scenes. Our method produces novel views with sharp edges and rich details, being

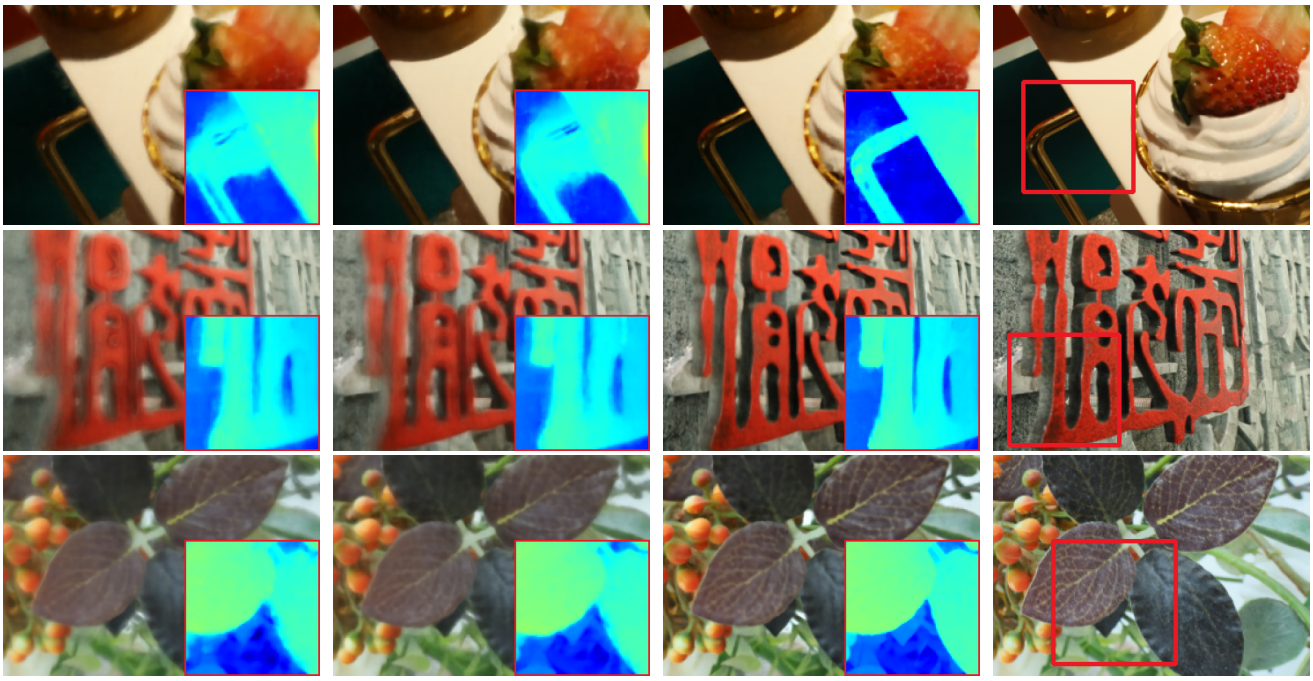



Figure 7. Qualitative comparison on real world defocus blur. The last column is captured for reference only and may be misaligned or have different exposures than ground truth.

the closest to ground truth. Previous methods demonstrate artifacts near the object boundary and blurry textures. The depth maps we predict are also sharper compared to other methods. Moreover, our method could produce more view consistent results than other baselines. And we provide additional results in conjunction with video output in the *supplementary material*.

# 6. Discussion and Conclusion

## 6.1. Why our framework works

Blindly recovering a sharp NeRF and the blur kernel simultaneously with only blurry images is an ill-posed problem, as the NeRF can also reconstruct a blurry scene that may "explain" the blurry images. Then how does our framework ensures that we get a sharp NeRF? As illustrated in the NeRF++ [54], the NeRF encodes priors for view consistent reconstruction. When the blurry input is view inconsistent, our framework compensates for the inconsistency using the DSK module, leading to the decomposition of the consistent sharp scene and the inconsistent blur pattern. Note that we define the blurry input as view consistent if they are equivalent to sharp observations of one blurry 3D scene. Blur in the real world is usually inconsistent. Each shot has a different blur pattern due to the randomness of camera movement or variability of focus distance. This can be further validated by the fact that when the *naive NeRF* reconstructs the real scene dataset, the results flicker severely when the viewpoint changes. Our framework addresses this issue, which is common in real world data.

## 6.2. Limitations

Our method can fail when the blur is view consistent, e.g., the camera coincidentally shaking in roughly the same direction across all views, or the camera having a fixed focal point (i.e., focuses on a single target). Deblurring a consistent blur can potentially be solved by introducing image priors, which we treat as future work. Our method may also fail when encounter input images that are severely blurred because the COLMAP may fail to reconstruct the camera poses. But in the experiments we find that this is only an issue in very blurry cases. For further discussion about such limitations, please refer to the *supplementary material*.

## 6.3. Conclusion

In this paper, we propose a simple but effective framework for training a sharp NeRF under blurry input. Experiments on both synthetic and real world scenes verify the effectiveness of our framework and demonstrate the significant improvement in quality over *naive NeRF* and image-space deblurring approaches. We hope that this work will further motivate research into NeRF-based approaches for deblurring applications.

**Acknowledgements.** Authors at HKUST and CityU were partly supported by the Hong Kong Research Grants Council (RGC), including the RGC Early Career Scheme under Grant 9048148 (CityU 21209119).

## References


[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021. 2

[2] Patrizio Campisi and Karen Egiazarian. *Blind image deconvolution: theory and applications*. CRC press, 2017. 2, 3

[3] Ayan Chakrabarti. A neural approach to blind motion deblurring. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 221–235, Cham, 2016. Springer International Publishing. 3

[4] Tony F Chan and Chiu-Kwong Wong. Total variation blind deconvolution. *IEEE transactions on Image Processing*, 7(3):370–375, 1998. 2

[5] Xiaogang Chen, Feng Li, Jie Yang, and Jingyi Yu. A theoretical analysis of camera response functions in image deblurring. In *European Conference on Computer Vision*, pages 333–346. Springer, 2012. 4

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[7] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv preprint arXiv:2103.14645*, 2021. 2

[8] Xin Huang, Qi Zhang, Feng Ying, Hongdong Li, Xuan Wang, and Qing Wang. Hdr-nerf: High dynamic range neural radiance fields. *arXiv preprint arXiv:2111.14451*, 2021. 2

[9] Tae Hyun Kim and Kyoung Mu Lee. Generalized video deblurring for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5426–5434, 2015. 3

[10] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Scholkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4038–4047, 2017. 3

[11] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5846–5854, 2021. 2

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[13] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR 2011*, pages 233–240. IEEE, 2011. 2

[14] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018. 3

[15] Junyong Lee, Hyeongseok Son, Jaesung Rim, Sunghyun Cho, and Seungyong Lee. Iterative filter adaptive network for single image defocus deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2034–2042, 2021. 3

[16] Anat Levin. Blind motion deblurring using image statistics. *Advances in Neural Information Processing Systems*, 19:841–848, 2006. 3

[17] Dongxu Li, Chenchen Xu, Kaihao Zhang, Xin Yu, Yiran Zhong, Wenqi Ren, Hanna Suominen, and Hongdong Li. Arvo: Learning all-range volumetric correspondence for video deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7721–7731, 2021. 3

[18] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv preprint arXiv:2104.06405*, 2021. 2

[20] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[21] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: Gan-based neural radiance field without posed camera. *arXiv preprint arXiv:2103.15606*, 2021. 2

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 3, 5

[23] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891, 2017. 3

[24] Seungjun Nah, Sanghyun Son, and Kyoung Mu Lee. Recurrent neural networks with intra-frame iterations for video deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8102–8111, 2019. 3

[25] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 2

[26] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. Motion deblurring in the wild. In *German conference on pattern recognition*, pages 65–77. Springer, 2017. 3

[27] Jinshan Pan, Haoran Bai, and Jinhui Tang. Cascaded deep video deblurring using temporal sharpness prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3043–3051, 2020. 3

[28] Chandramouli Paramanand and Ambasamudram N Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1115–1122, 2013. 3

[29] Haesol Park and Kyoung Mu Lee. Joint estimation of camera pose, depth, deblurring, and super-resolution from a blurred image sequence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4613–4621, 2017. 3

[30] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2

[31] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[33] Jiayan Qiu, Xinchao Wang, Stephen J Maybank, and Dacheng Tao. World from blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8493–8504, 2019. 3

[34] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *arXiv preprint arXiv:2103.13744*, 2021. 2

[35] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992. 2

[36] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[37] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 6

[38] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020. 2

[39] Hyeongseok Son, Junyong Lee, Sunghyun Cho, and Seungyong Lee. Single image defocus deblurring using kernel-sharing parallel atrous convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2642–2650, 2021. 3, 7

[40] Hyeongseok Son, Junyong Lee, Jonghyeop Lee, Sunghyun Cho, and Seungyong Lee. Recurrent video deblurring with blur-invariant motion estimation and pixel volumes. *ACM Transactions on Graphics (TOG)*, 40(5):1–18, 2021. 3, 7

[41] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1279–1288, 2017. 3

[42] Yu-Wing Tai, Ping Tan, and Michael S Brown. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1603–1618, 2010. 3

[43] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 3

[44] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182, 2018. 3

[45] Phong Tran, Anh Tuan Tran, Quynh Phung, and Minh Hoai. Explore image deblurring via encoded blur kernel space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11956–11965, 2021. 3

[46] Ruxin Wang and Dacheng Tao. Recent progress in image deblurring. *arXiv preprint arXiv:1409.6838*, 2014. 2, 3, 4

[47] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2

[48] Patrick Wieschollek, Michael Hirsch, Bernhard Scholkopf, and Hendrik Lensch. Learning blind motion deblurring. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 231–240, 2017. 3

[49] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1107–1114, 2013. 2

[50] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 5

[51] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[52] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14821–14831, 2021. 3, 7

[53] Kaihao Zhang, Wenhan Luo, Yiran Zhong, Lin Ma, Bjorn Stenger, Wei Liu, and Hongdong Li. Deblurring by realistic blurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2737–2746, 2020. 3

[54] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 8

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

[56] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020. 3