Deblur-NSFF: Neural Scene Flow Fields for Blurry Dynamic Scenes

Achleshwar Luthra[1*] Shiva Souhith Gantha[2†] Xiyun Song[3] Heather Yu[3]
Zongfang Lin[3] Liang Peng[3]
[1]*Carnegie Mellon University* achleshl@andrew.cmu.edu
[2]*Georgia Institute of Technology* sgantha3@gatech.edu
[3]*Futurewei Technologies* {xsong; hyu; zlin1; lpeng}@futurewei.com

Abstract

In this work, we present a method to address the problem of novel view and time synthesis of complex dynamic scenes considering the input video is subject to blurriness caused due to camera or object motion or out-of-focus blur. Neural Scene Flow Field (NSFF) has shown remarkable results by training a dynamic NeRF to capture motion in the scene, but this method is not robust to unstable camera handling which can lead to blurred renderings. We propose Deblur-NSFF, a method that learns spatially-varying blur kernels to simulate the blurring process and gradually learns a sharp time-conditioned NeRF representation. We describe how to optimize our representation for sharp space-time view synthesis. Given blurry input frames, we perform both quantitative and qualitative comparison with state-of-the-art methods on modified NVIDIA Dynamic Scene dataset. We also compare our method with Deblur-NeRF, a method that has been designed to handle blur in static scenes. The demonstrated results show that our method outperforms prior work.

1. Introduction

There has been a recent wave of NeRF-based methods for free-viewpoint rendering of static 3D scenes with impressive results. Even more recently, the research community has shifted its focus towards novel space-time synthesis of dynamic scenes that comprises moving objects such as people or pets. This is sometimes also referred to as *free-viewpoint video* and this enables plenty of applications such as cinematic effects like bullet-time visual effect (as shown in "The Matrix") from monocular videos, free-viewpoint selfies [23], background-foreground separation [32] (where foreground refers to objects in motion), virtual 3D teleportation [22] and so on.

Novel view synthesis for a dynamic scene is a challenging task. This requires expensive and arduous setups of multiple-camera capturing rigs which are impractical to scale. There can be ambiguous solutions as multiple scene settings can lead to the same observed image sequences and additionally, moving objects also add to the difficulty of this problem statement. Recent methods [5, 8, 9, 23, 24, 29] have proposed unique solutions to deal with the task of dynamic novel view-time synthesis but they still have many limitations. Methods like Nerfies [23] and HyperNeRF [24] learn a static canonical model and handle deformations via a separate *ray bending* network whereas methods like NSFF [8] and DVS [5] learn correspondences over time (or temporal consistency) via 3D *scene flow* and display the ability to handle larger motion (compared to *ray bending* network-based methods) in the scene. However, these methods are not designed to handle unstable camera motion hence preventing their deployment in real world scenarios where blur is very common.

Although NeRF variants have achieved remarkable success in space-time view synthesis, these methods require carefully captured videos or images with well-calibrated camera parameters hence limiting them to controlled environments. In the past, only a few methods have sufficiently addressed the problem of novel view synthesis from images subject to motion or out-of-focus blur. To the best of our knowledge, Deblur-NeRF [14] is the first method that can reconstruct sharp NeRF from blurry inputs. Deblur-NeRF proposed a deformable sparse kernel estimation module to simulate the blurring process while training a NeRF model. More recently, DP-NeRF [7] introduced a rigid blurring kernel that utilizes physical scene priors, and an adaptive weight refinement scheme that considers relationship between depth and blur to render realistic results. These methods achieve impressive results but they are limited to static scenes and cannot handle moving objects in the scene. On the contrary, our proposed method can construct a sharp NeRF while also handling dynamic scenes.

In this paper, we take best of both the worlds and pro-

[*]work done during an internship at Futurewei Technologies
[†]work done during an internship at Futurewei Technologies

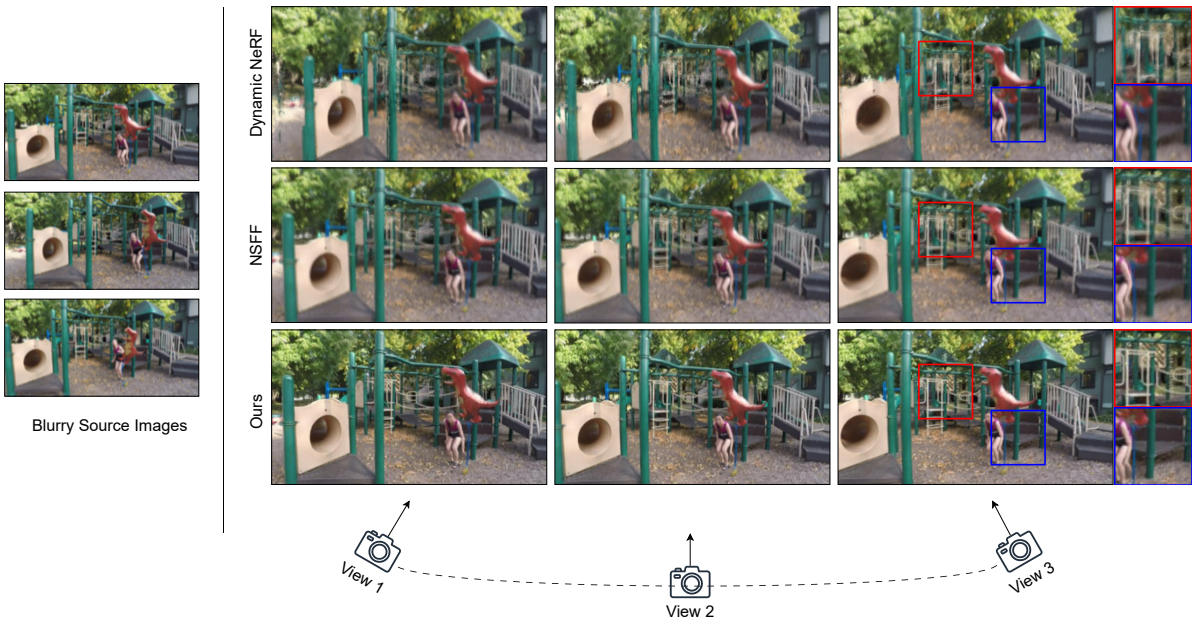


Figure 1. Given a set of blurry source images of a *dynamic* scene, previous methods struggle to reconstruct sharp renderings of the scene as shown in the first two rows. Our method is able to produce sharp outputs by recovering the details in the scene. On the rightmost side, the red boxes show that our method is able to reconstruct a much sharper static background scene compared to DVS (or Dynamic NeRF) [5] and NSFF [8] as well as the dynamic region shown in the blue boxes.

pose Deblur-NSFF, a method that learns a continuous volumetric function mapping a 3D location, direction and time to reflectance, density and 3D scene motion. We train a sharp NSFF model by maintaining temporal consistency using scene flow fields warping loss and by simulating the blurring process using a Kernel Estimation Network. During inference time, our algorithm is able to generate sharp novel space-time renderings from blurry input frames. Our work tackles a novel problem that has not yet been covered in the previous research, to the best of our knowledge. We also perform ablation experiments to test specific components of our method and provide detailed analysis of our proposed algorithm.

In summary, the key contributions of this work include:

- We introduce an original problem statement that is vital to the real-world deployment of NeRF variants for dynamic scenes and we also propose Deblur-NSFF, a method that can perform novel space-time synthesis from videos subject to motion or out-of-focus blur.
- We provide quantitative and qualitative comparison of our method with previous state-of-the-art methods on modified NVIDIA Dynamic Scenes Dataset [35] and show that our method outperforms prior work. We also discuss how we generated the modified dataset.
- Through our extensive experiments, we analyse importance of different hyperparameters ($i.e.$ kernel points) and different components of our method.
- We also provide a qualitative comparison of Deblur-NSFF with Deblur-NeRF in Sec. 4.3.3 and demonstrate the importance of our proposed method.

2. Related Work

Neural Radiance Fields for Static Scenes: There have been numerous methods that build the 3D scene geometry using point clouds, parametric surfaces and meshes and use it to render novel views. Recently, NeRF [19], proposed by Mildenhall *et al*., introduced a neural network that takes a 5D input to model the continuous 3D scene by using differentiable volumetric rendering techniques. Success of NeRF inspired many subsequent works on improving upon different aspects and limitations of NeRF including the photometric and geometric quality [2, 3, 18, 30, 34], training and rendering efficiency [6, 12, 20, 27] and reducing the number of input views used [3, 21, 26, 36]. All these works assume ideal input views that are perfectly captured.

There are few works that have explored non-ideal input, for example [10, 17, 31] consider the case where the camera poses are not available. [16] disentangles the lighting with the geometry which helps when the input views have some inconsistencies. [2] helps in removing aliasing artifacts as the input images are scaled.

Neural Radiance Fields for General Dynamic Scenes: Compared to static scenes, this is an even more challeng-

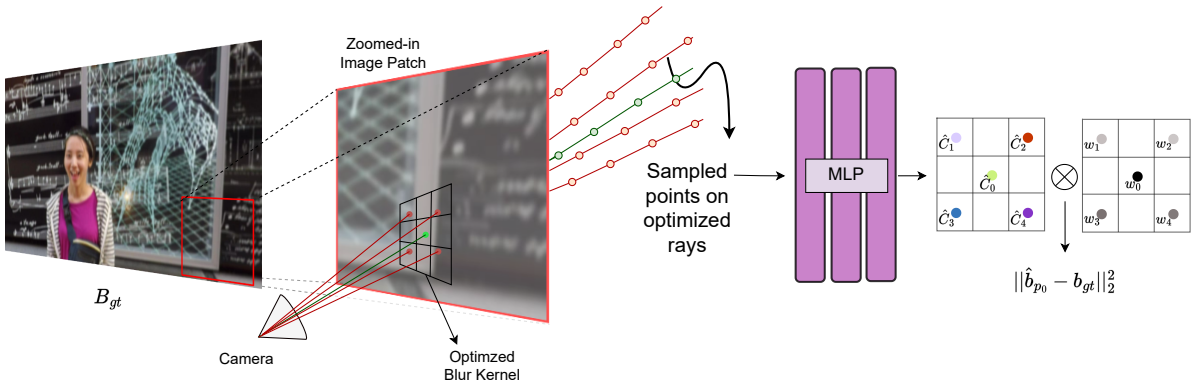


Figure 2. Visual illustration of our proposed methodology. Please find the details in Sec. 3.2. Note that b represents a single pixel in B $i.e.$ blurry image. The image used in this diagram has been taken from NVIDIA Dynamic Scenes Dataset [35]

ing problem because along with capturing view-dependent ambiguities in geometry and appearances, a network trained for dynamic scene also needs to model deformations in geometry with varying time. There have been some recent NeRF-based successful attempts for the task of 4D reconstruction in space and time, given real monocular RGB videos [1, 8, 9, 11, 13, 23, 24, 29, 33, 35] or synthetic videos [4, 25].

Tewari *et al*. [28] categorizes these approaches into: **(a)** time-varying NeRFs and **(b)** controllable NeRFs that use ray-bending technique to handle deformations. Time-varying NeRFs are conditioned on encoded time-input [8, 9, 13, 33] and assume no prior knowledge of the 3D scene. These methods rely on learning from different data modalities such as depth, optical flow, segmentation masks, and camera poses. In addition to data-based learning, these methods also take advantage of geometric regularization losses to maintain consistency across time. Although these methods have shown impressive results and can empirically handle larger motion, they entangle geometry, appearance and deformation whereas controllable NeRFs [23–25, 29] can disentangle deformation from geometry and appearance by modelling a separate canonical model to map from deformed space to canonical space but cannot capture larger range of motion.

Our focus in this work has been to capture larger range of motion when the input images are blurry thus our ideas have been majorly inspired from Neural Scene Flow Fields (NSFF) [8].

Neural Radiance Fields from Blurry Images: The methods that we have discussed so far have shown quality results when the input images are well captured without any artifacts, but they produce undesirable outputs when the images are exposed to artifacts such as camera motion blur. A recent method called Deblur-NeRF [14] deals with blurry input images by learning a Deformation Sparse Kernel module that models the spatially-varying blurring process. Another method - DP-NeRF [7] also follows a similar approach by constraining their blur kernel with physical scene priors derived from the blurring process during image acquisition. These methods are limited in their approach to static scenes and to the best of our knowledge, none has considered dealing with blurry dynamic scenes. Our work extends the ideas from Deblur-NeRF [14] to non-rigid scenes and we also show a comparison with [14] through our experiments in Sec. 4.3.3.

3. Our Methodology

In the section, we discuss our proposed solution in detail. First, we cover Neural Scene Flow Fields in Sec. 3.1 to keep this work self-contained. Then, in Sec. 3.2 and Sec. 3.3, we describe our algorithm and kernel estimation network respectively. Following this, we talk about how we optimize our network in Sec. 3.4.

3.1. Background: Neural Scene Flow Fields

To represent a dynamic scene, Neural Scene Flow Fields (also known as NSFF [8]) extend the idea of Neural Radiance Fields (NeRFs) by modelling 3D motion as dense scene flow fields. It learns a combination of static and dynamic NeRFs. Static NeRF is a time-independent multi-layer perceptron (MLP), denoted by F_θ^{st}, that takes as input a position ($\mathbf{x}$) and viewing direction ($\mathbf{d}$), and outputs RGB color ($\mathbf{c}$), volumetric density (σ), and unsupervised 3D blending weights ($\mathbf{v}$) that determines how to blend RGBσ from static and dynamic representation:

$$(\mathbf{c}, \sigma, \mathbf{v}) = F_\theta^{st}(\mathbf{x}, \mathbf{d}) \tag{1}$$

Dynamic NeRF, denoted by F_θ^{dy}, explicitly models a view-dependent as well as time-dependent MLP that takes an additional input, $i.e.$, time ($\mathbf{t}$) along with $\mathbf{x}$ and $\mathbf{d}$. On top of color and density, the model also predicts forward and

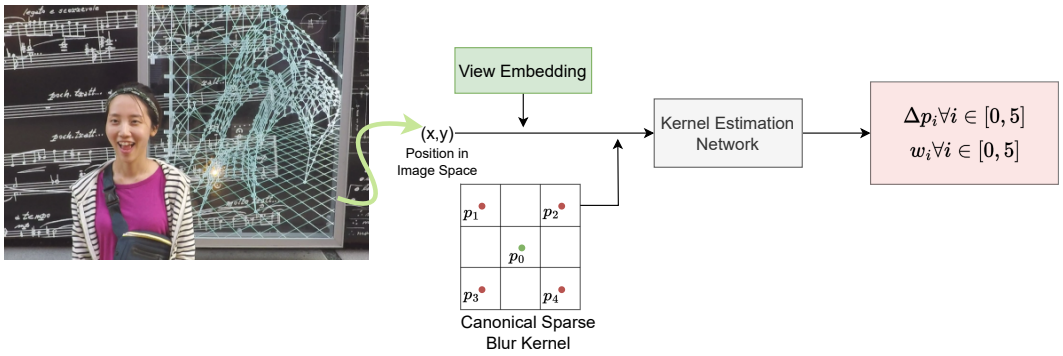


Figure 3. **Kernel Estimation Network:** We train this netowork to learn generalizable spatially-varying blur kernels. The details can be found in Sec. 3.3. The image used in this diagram has been taken from NVIDIA Dynamic Scenes Dataset [35]

backward 3D scene flow $\mathcal{F}_t = (\mathbf{f}_{t\to t+1}, \mathbf{f}_{t\to t-1})$ and disocclusion weights $\mathcal{W}_t = (\mathbf{w}_{t\to t+1}, \mathbf{w}_{t\to t-1})$ to tackle motion disocclusions in 3D space (described in detail in Section 3.4):

$$(\mathbf{c}_t, \sigma_t, \mathcal{F}_t, \mathcal{W}_t) = F_\theta^{dy}(\mathbf{x}, \mathbf{d}, \mathbf{t}) \tag{2}$$

where $\mathbf{c}_t$ and σ_t denote color and volumetric density for point $\mathbf{x}$ at time $\mathbf{t}$. Final color value is estimated using the blending weights as per the following rendering equations:

$$\hat{\mathbf{C}}_t^{cb}(r_t) = \int_{z_n}^{z_f} \mathbf{T}_t^{cb}(z)\sigma_t^{cb}(z)\mathbf{c}_t^{cb}(z)dz \tag{3}$$

where

$$\sigma_t^{cb}(z)\mathbf{c}_t^{cb}(z) = \mathbf{v}(z)\mathbf{c}(z)\sigma(z) + (1-\mathbf{v}(z))\mathbf{c}_t(z)\sigma_t(z) \tag{4}$$

and $\mathbf{T}_t$ denotes transmittance at time t, z_n and z_f denote near depth and far depth along a ray. This output $\hat{\mathbf{C}}_t^{cb}(r_t)$ is finally trained against $\mathbf{C}_t(r_t)$, *i.e.*, the ground truth RGB color value at the pixel corresponding to ray r_t:

$$\mathcal{L}_{cb} = ||\hat{\mathbf{C}}_t^{cb}(r_t) - \mathbf{C}_t(r_t)||_2^2. \tag{5}$$

3.2. Deblur-NSFF

We take the idea of NSFF [8] but instead of rendering a single ray per pixel, we consider N rays (N = 5 as per Fig. 2) for each pixel. This is done in order to simulate the following blurring process:

$$\mathrm{b}(x,y) = \mathrm{c}(x,y) \circledast \mathrm{h} \tag{6}$$

where h is blur kernel, $\mathrm{c}(x,y)$ is sharp image that we want our MLP to learn, $\mathrm{b}(x,y)$ is blurred image, and $\circledast$ denotes convolution operator. h is a $\mathrm{K}\times\mathrm{K}$ window and ideally N should be equal to K^2 but since we are training a NeRF model to obtain $c(x,y)$, it would be infeasible to project K^2 rays as it will shoot up the memory requirements and training time. Hence, we fix $\mathrm{K} = 3$ (kernel window) with $\mathrm{N} = 5$ rays in our experiments. In our ablation experiments (Sec. 4.3.2), we have shown quantitative comparison by altering N and we show that setting $\mathrm{N} = \mathrm{K}^2$ yields insignificant boost in results.

Once we have the optimized N rays, our algorithm estimates $\hat{C}_t^{cb}(r_{t_i})\forall i \in [0,\mathrm{N})$ as per Eq. (3). Each pixel now has N associated color values and a sparse blur kernel h with weights $w_i \forall i \in [0,\mathrm{N})$ predicted by Kernel Estimation Network (KernelNet) as shown in Fig. 3. Using these values, we simulate the blurring process as per Eq. (6) and obtain $\hat{\mathbf{B}}_t(r_t)$ where r_t corresponds to a pixel at time t and $\hat{\mathbf{B}}_t$ denotes output blurry image at time t. This is further supervised using the ground truth blurry image $\mathbf{B}_{gt}$:

$$\mathcal{L}_{mse-blur} = ||\hat{\mathbf{B}}_t(r_t) - \mathbf{B}_{gt}(r_t)||_2^2. \tag{7}$$

As per Eq. (2), F_θ^{dy} outputs $\mathcal{F}_t$ for all the points in 3D space. In our work, we get forward and backward flows for each sampled point on all the optimized rays. Using $\mathcal{F}_t$, we can offset each point to a neighboring frame t' (where $t' \in \{t-1, t+1\}$) and volume render with associated color $(c_{t'})$ and density $(\sigma_{t'})$. This shall give us a rendered image of time t' warped to time t, denoted as:

$$\hat{\mathbf{C}}_{t'\to t}(r_t) = \int_{z_n}^{z_f} \mathbf{T}_{t'}(z)\sigma_{t'}(\mathbf{r}_{t\to t'}(z))\mathbf{c}_{t'}(\mathbf{r}_{t\to t'}(z), \mathbf{d_t})dz \tag{8}$$

where

$$\mathbf{r}_{t\to t'}(z) = \mathbf{r}_t(z) + \mathbf{f}_{t\to t'}(\mathbf{r}_t(z)). \tag{9}$$

Instead of $\hat{\mathbf{C}}_{t'\to t}(r_t)$, we obtain $\hat{\mathbf{C}}_{t'\to t}(r_{t_i})\forall i \in [0,\mathrm{N})$ and repeat the blurring process using the same blur kernel for the pixel corresponding to $\mathbf{r}_t$. This gives us warped rendered blurry view - $\hat{\mathbf{B}}_{t'\to t}(r_t)$ and then our goal is to minimize:

$$\mathcal{L}_{pho} = ||\hat{\mathbf{B}}_{t'\to t}(r_t) - \mathbf{B}_{gt}(r_t)||_2^2. \tag{10}$$

More details on temporal photometric consistency loss (Eq. (10)) are discussed in Sec. 3.4.

3.3. Kernel Estimation Network

As shown in Fig. 2, we render N rays for each pixel and those optimized rays are determined according to the predicted blur kernel h. The blur kernel is obtained using Kernel Estimation Network (KernelNet), as visually illustrated in Fig. 3. Inspired by [14], we use MLP, denoted by G_ϕ, as a design choice for KernelNet. G_ϕ takes a query pixel $\mathbf{p}$, a canonical kernel h', and a view embedding $\mathbf{l}$ as inputs, and outputs $\Delta_{\mathbf{p}_i}$ and $w_i \forall i \in [0, \mathrm{N})$ where $\mathrm{N} \leq \mathrm{K}^2$:

$$(\Delta_{\mathbf{p}_i}, w_i) = G_\phi(\mathbf{p}, \mathrm{h}', \mathbf{l}) \tag{11}$$

where $\Delta_{\mathbf{p}_i}$ is the offset for each $\mathbf{p}_i$ which are pre-defined positions on the canonical kernel h', and w_i is their associated weight. Note that in Eq. (11), we want our network to learn optimized blur kernel for each pixel as blurring is a spatially-varying process, as well as it usually varies with the viewing direction [14], which justifies the need for a view-embedding.

To confine the solution obtained from KernelNet, we define a *boundary constant* ϵ (= 0.1 in our experiments) and multiply it with the outputs from Eq. (11). This helps us keep the neighboring points, that will be considered in producing the final blur color for each pixel, closer to the pixel in consideration.

Inference: During inference, we want to render sharp novel views. Our model is trained with the following hypothesis: while KernelNet is responsible for simulating the blurring process, the static and dynamic NeRFs learn a sharp scene. This implies that while testing, we can get rid of KernelNet and use trained sharp NeRFs to generate novel space-time views. Taking advantage of the splatting-based plane-sweep volume rendering approach [8], we have rendered novel views at fixed times, novel times at fixed views, and space-time interpolation. Further comparison with different approaches has been shown in Fig. 4 and discussed in detail in Sec. 4.

3.4. Optimization

Temporal photometric consistency. As introduced in [8], this loss ensures the consistency of the scene at time t with the adjacent times t' after accounting for the motion due to 3D scene flow. We warp the scene from time t' to t using the 3D scene flow estimation output from the Dynamic NeRF (F_θ^{dy}) which ensures that any motion that occurred in that time period is undone. We use the warped point locations $\mathbf{x_{t'}}$ on the ray $\mathbf{r_{t'}}$ to query the associated color ($c_{t'}$) and density ($\sigma_{t'}$) at time t'. Using the color and density information we render the image with $\hat{\mathbf{C}}_{t'\to t}(r_t)$ as shown in equation [8]. The blur kernel for the the ray r_t corresponding to time t is then applied to get the blurry frame $\hat{\mathbf{B}}_{t'\to t}(r_t)$.

The calculation of loss over regions which get disoccluded due to motion in the scene is ambiguous. To help with this, we let the Dynamic NeRF (F_θ^{dy}) also output disocclusion weights $\mathcal{W}_t = (\mathbf{w}_{t\to t+1}, \mathbf{w}_{t\to t-1}) \in [0, 1]$. The weights $(\mathbf{w}_{t\to t+1}, \mathbf{w}_{t\to t-1})$ decide the contribution of the temporal photometric consistency loss at each location in the scene to the total loss. To calculate $\mathcal{W}_t$, we first volume render the weights along a ray $\mathbf{r_t}$ using the density values from time t'. We then use this accumulated weight ($\hat{W}_{t'\to t}$) for each 2D pixel as the weightage of the temporal photometric consistency loss as shown in the equations:

$$\hat{W}_{t'\to t}(\mathbf{r}_t) = \int_{z_n}^{z_f} \mathbf{T}_{t'}(z)\, \sigma_{t'}(\mathbf{r}_{t\to t'}(z))\, w_{t\to t'}(\mathbf{r}_t(z)) dz \tag{12}$$

To avoid the trivial solution, we add l_1 regularization to force the dissocclusion weights to be near 1, resulting in the following equation for the loss:

$$\mathcal{L}_{pho} = \sum_{\mathbf{r}_t} \sum_{t'\in\mathcal{N}(t)} \hat{W}_{t'\to t}(\mathbf{r}_t) ||\hat{\mathbf{B}}_{t'\to t}(\mathbf{r}_t) - \mathbf{B}_t(\mathbf{r}_t)||_2^2 + \lambda \sum_{\mathbf{x}_t} ||w_{t\to t'}(\mathbf{x}_t) - 1||_1, \tag{13}$$

where λ is a regularization parameter which is kept as 0.1 as in [8].

Alignment Loss: We need to add further regularization to train KernelNet as well as gear NeRF model towards learning a sharp representation of the scene. Without any constraints, the NeRF model together with the optimized kernel might learn to map well to blurry ground truth image but during inference, we may have some unexpected distortions because we produce renderings using NeRF model without KernelNet. To avoid this, the sparse kernel is initialized such that the optimized rays are close to the input ray and the kernel weights corresponding to the optimized rays are similar to gaussian kernel representation. This will ensure that when we start the training, all the kernel points are near the pixel centers. Further, one of the optimized rays is forced to be close to the input ray as shown in the equation below:

$$\mathcal{L}_{align} = ||\mathbf{q_0} - \mathbf{p}|| \tag{14}$$

where $\mathbf{q_0}$ corresponds to a fixed index in the output from KernelNet. In a nutshell, the overall loss function is:

$$\mathcal{L} = \mathcal{L}_{mse-blur} + \mathcal{L}_{pho} + \lambda_{align}\mathcal{L}_{align} \tag{15}$$

here we keep $\lambda_{align} = 0.1$.

Method	Dynamic Only			Full		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
DynamicNeRF	17.22	0.36	0.39	20.61	0.49	0.39
NSFF	20.76	0.59	0.31	24.15	0.70	0.32
Ours	**21.14**	**0.64**	**0.22**	**25.19**	**0.79**	**0.23**

Table 1. Quantitative comparison for **complete** NVIDIA Dynamic Scene Dataset [35]. Metrics reported here are calculated by taking an average along all the scenes. Metrics for individual scenes are reported in Tab. 2.

Scene	PSNR (↑)			SSIM (↑)			LPIPS (↓)		
	DVS	NSFF	Ours	DVS	NSFF	Ours	DVS	NSFF	Ours
Playground	17.33	20.91	**22.52**	0.70	0.58	**0.72**	0.44	0.41	**0.27**
Jumping	22.88	25.61	**26.05**	0.65	0.79	**0.83**	0.26	0.22	**0.15**
Balloon 1	16.59	22.04	**22.92**	0.34	0.65	**0.74**	0.48	0.35	**0.24**
Balloon 2	21.08	25.29	**25.71**	0.47	0.70	**0.80**	0.41	0.38	**0.24**
DynamicFace	13.82	18.50	**20.50**	0.28	0.63	**0.78**	0.51	0.34	**0.25**
Skating	24.99	30.16	**31.72**	0.68	0.86	**0.91**	0.25	0.21	**0.14**
Truck	26.20	27.22	**28.54**	0.74	0.79	**0.84**	0.33	0.28	**0.18**
Umbrella	22.01	23.50	**23.58**	0.44	0.60	**0.66**	0.50	0.38	**0.31**

Table 2. Quantitative comparison for **each scene** in NVIDIA Dynamic Scenes Dataset [35] with DVS [5] and NSFF [12].

Scene	PSNR (↑)			SSIM (↑)			LPIPS (↓)		
	DVS	NSFF	Ours	DVS	NSFF	Ours	DVS	NSFF	Ours
Playground	15.19	18.61	**19.00**	0.30	0.56	**0.62**	0.41	0.35	**0.24**
Jumping	17.43	**19.59**	19.58	0.40	0.56	**0.58**	0.30	0.27	**0.21**
Balloon 1	13.58	**18.48**	18.35	0.19	0.46	**0.47**	0.61	0.34	**0.26**
Balloon 2	18.90	22.44	**23.02**	0.34	0.62	**0.71**	0.37	0.35	**0.21**
DynamicFace	13.40	22.01	**23.25**	0.16	0.75	**0.82**	0.45	0.29	**0.15**
Skating	18.25	21.69	**22.39**	0.46	0.67	**0.72**	0.31	0.28	**0.19**
Truck	24.42	26.26	**26.88**	0.67	0.76	**0.79**	0.29	0.26	**0.17**
Umbrella	16.66	**17.02**	16.68	0.34	0.41	**0.41**	0.44	0.36	**0.30**

Table 3. Quantitative comparison on **dynamic** regions for **each scene** in Dynamic Scenes Dataset [35] with DVS [5] and NSFF [12].

4. Experimental Details & Analysis

Dataset Generation: To test our proposed methodology, we have used NVIDIA Dynamic Scene Dataset [35] and added synthetic blur to it. We have used *Gaussian Blur* by torchvision [15] to introduce blur to random frames by fixing kernel size to 3×3 and taking random values of sigma as it helps to randomize the blur kernel used for each image. We have randomly selected $\approx 80\%$ of the frames and blurred them for training our models. During evaluation, we use ground truth sharp images against the synthesized results by our trained network.

4.1. Quantitative Analysis

Similar to NSFF [8], we also consider 24 frames per scene corresponding to single camera for training and use remaining 11 held-out camera views per time frame for evaluation. We have reported PSNR, SSIM, and LPIPS metrics values for novel-view synthesis for the entire dataset in Tab. 1 and scene-wise values in Tab. 2. Our approach outperforms prior work in all the considered metrics for both dynamic regions as well as the entire scene. Even for scene-wise comparison, our methods performs better on all the scenes in the Dynamic Scenes dataset [35]. Compared to NSFF (second best method), Deblur-NSFF achieves an improvement of 4.3%, and 12.8%, in PSNR and SSIM respectively, and a decrease of 28.12% in LPIPS value for *full* scene.

4.2. Qualitative Analysis

We have shown qualitative comparison with DVS [5] and NSFF [8] in Fig. 4. In each scene, we have selected one static and one dynamic region. DVS and NSFF both do a great job in capturing the motion in scene by treating the dynamic regions of the scene as time-dependent and view-dependent effects but these methods are not specifically designed to handle blur in the scene. Our method can produce sharp results even when the model is trained on blurry input frames. As shown in second row of Fig. 4, in red box, our method captures sharper background and we can also better see the teeth of the balloon whereas in blue box, we can see the text "open" much clearer as compared to DVS

Figure 4. Qualitative Comparison of our method with DVS [5] and NSFF [12]. Please check description in Sec. 4.2.

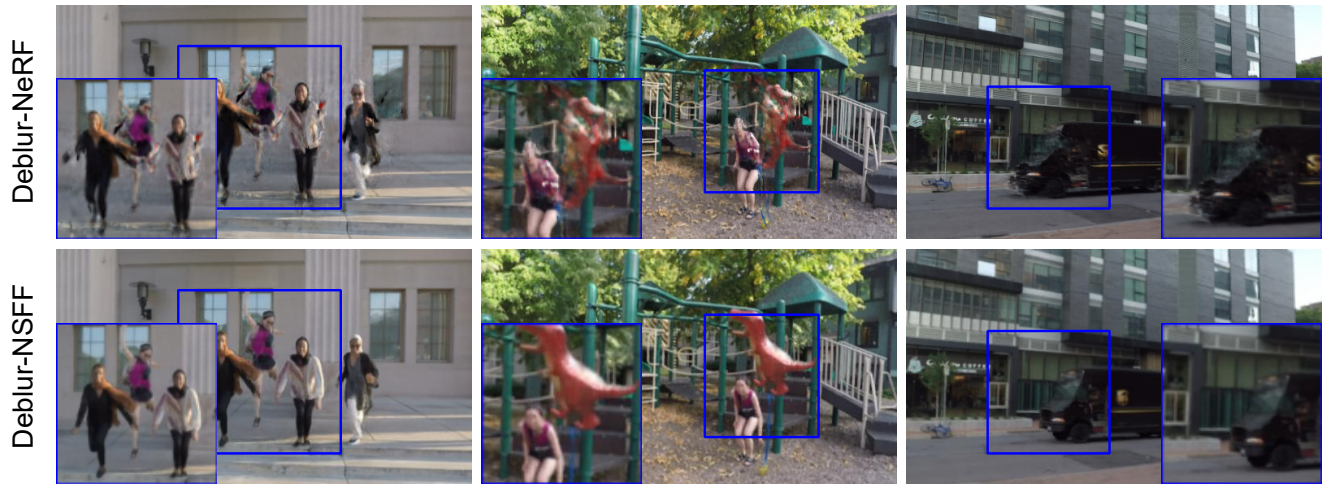


Figure 5. Qualitative Comparison of our method with Deblur-NeRF [14]. Please check description in Sec. 4.3.3.

and NSFF. In third row (in Fig. 4) the lines of the stairs are sharpest in our results whereas in DVS and NSFF they are relatively smoothened. In blue box, we can see some artifacts in the reflection (of what looks like a tree) which are also resolved in our results and other details of the window are also more perceivable. For space-time view synthesis, we used the same approach as NSFF during evaluation. For better comparison, it is recommended to watch the supplementary videos.

4.3. Ablation Experiments

4.3.1 Camera Origins

As stated in [14], Eq. (6) is a close approximation of the blurring process but it assumes that while capturing a pixel $\mathbf{p}$, the camera origin $\mathbf{o}$ remains the same for all the neighboring pixels that fall within the kernel window $\mathbf{h}$. Inspired by the physical blurring process, we include optimizing for camera origins corresponding to each pixel as part of our ablation experiments. This basically requires changes in Eq. (11) which now also outputs $\Delta_{\mathbf{o}}$:

$$(\Delta_{\mathbf{o}_i}, \Delta_{\mathbf{p}_i}, w_i) = G_\phi(\mathbf{p}, \mathrm{h}', \mathbf{l}) \tag{16}$$

As shown in Fig. 2, we project *optimizedrays* from a single camera origin. Instead of fixing a single camera origin corresponding to an optimized blur kernel, we now take $\mathbf{N}$ camera origins. We have shown a quantitative comparison of this approach with our proposed baseline in Tab. 4. Shifting camera origins gives us marginal improvement in performance while also increasing the training time by approximately 1 hour for reach 10k iterations.

Scene	PSNR (↑)		SSIM (↑)		LPIPS (↓)	
	Ours w/o shift	Ours w/ shift	Ours w/o	Ours w/	Ours w/o	Ours w/
Playground	22.52	22.54	0.72	0.73	0.27	0.26
Jumping	26.05	26.06	0.83	0.84	0.15	0.16
Balloon 1	22.92	22.91	0.74	0.73	0.24	0.25
Balloon 2	25.71	25.72	0.80	0.80	0.24	0.24
DynamicFace	20.50	20.50	0.78	0.77	0.25	0.24
Skating	31.72	31.73	0.91	0.92	0.14	0.14
Truck	28.54	28.53	0.84	0.85	0.18	0.17
Umbrella	23.58	23.57	0.66	0.66	0.31	0.30

Table 4. **Ablation experiments:** Quantitative comparison for **each scene** in NVIDIA Dynamic Scene Dataset. w/ means our method with origin shift as explained in Sec. 4.3.1 and w/o means without, *i.e.*, our baseline method.

4.3.2 Kernel Points

In Fig. 3, we have shown 4 neighboring points for a pixel of interest $\mathbf{p}_0$. To study the importance of number of kernel points, Deblur-NeRF [14] ablates the value of this hyper-parameter and concluded that setting $\mathbf{N} = 5$ yields optimal performance. Increasing $\mathbf{N}$ beyond 5 gives minor improvement while significantly shooting up the training time and computational memory requirements. Decreasing $\mathbf{N}$ less than 5 notably affects the performance with respect to both SSIM and PSNR values. To test this, we train our method by setting $\mathbf{N} = 9$ and report the results obtained in Tab. 5

Scene	PSNR (↑)		SSIM (↑)		LPIPS (↓)	
	$\mathbf{N} = 5$	$\mathbf{N} = 9$	$\mathbf{N} = 5$	$\mathbf{N} = 9$	$\mathbf{N} = 5$	$\mathbf{N} = 9$
Playground	22.52	22.60	0.72	0.75	0.27	0.26

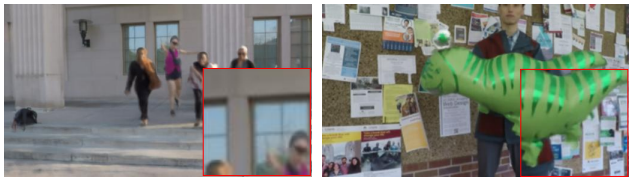
Table 5. **Ablation experiments:** Quantitative comparison for different values of $\mathbf{N}$.

4.3.3 Comparison with Deblur-NeRF

This problem of 3D novel-view synthesis of dynamic scenes where the frames are subject to blur is rather under-explored topic. To the best of our knowledge, only Deblur-NeRF deals with this problem but that too specifically for static scenes. More recently, DP-NeRF [7] also focused on the same problem statement for static scenes. Although Deblur-NeRF has been designed to deal with static scenes and it is expected to fail when tested on dynamic scenes, we still show its comparison with our method to stress upon the relevance of our work to capture motion while sharpening the scene.

In Fig. 5, we have shown qualitative comparison with Deblur-NeRF [14] on blurred scenes from NVIDIA Dynamic Scenes dataset. Although Deblur-NeRF does a great job in sharpening the static regions in the scene, it shows clear artifacts in the dynamic regions. In contrast, our method is able to recover full scene.

(a) Our method is unable to recover minute details such as face expressions.



(b) Our method produces artifacts during time interpolation

Figure 6. Limitations of our method and potential for future work.

5. Final Discussion

5.1. Limitation

Our method is not able to recover minute details in the scene such as facial definition of a person, as shown in Fig. 6a. Our method does a great job at space interpolation but the results show artifacts when we do time interpolation or space-time interpolation. We have shown these results in Fig. 6b. These outputs are currently obtained using splatting-based plane-sweep volume rendering approach [8] for time-interpolation and as per our results, we can say that there's a scope of improvement here.

5.2. Conclusion

To summarize, we have proposed an effective solution to render sharp novel view and time synthesis of dynamic scenes from blurry inputs. Our extensive experiments on NVIDIA Dynamic Scenes dataset solidify our proposals and signify the importance of our method. We believe that handling blur for sharp space-time interpolation is rather under-explored area and we hope that our work will encourage future research in this field.

References

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. In *Neural Information Processing Systems*, 2021. 3

[2] 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. *ICCV*, 2021. 2

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 2

[4] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314, 2020. 3

[5] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 1, 2, 6, 7

[6] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 2

[7] Dogyoon Lee, Minhyeok Lee, Chajin Shin, and Sangyoun Lee. Dp-nerf: Deblurred neural radiance field with physical scene priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12386–12396, June 2023. 1, 3, 8

[8] 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 (CVPR)*, 2021. 1, 2, 3, 4, 5, 6, 8

[9] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. *ArXiv*, abs/2211.11082, 2022. 1, 3

[10] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[11] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *ArXiv*, abs/2205.15723, 2022. 3

[12] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2, 6, 7

[13] Y. Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. *ArXiv*, abs/2301.02239, 2023. 3

[14] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-nerf: Neural radiance fields from blurry images. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12851–12860, 2021. 1, 3, 5, 7, 8

[15] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 6

[16] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 2

[17] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[18] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR*, 2022. 2

[19] 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 *ECCV*, 2020. 2

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 2

[21] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[22] Sergio Orts, Christoph Rhemann, Sean Fanello, David Kim, Adarsh Kowdle, Wayne Chang, Yury Degtyarev, Philip Davidson, Sameh Khamis, Minsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip Chou, Sarah Mennicken, Julien Valentin, Pushmeet Kohli, Vivek Pradeep, Shenlong Wang, and Shahram Izadi. Holoportation: Virtual 3d teleportation in real-time. 10 2016. 1

[23] Keunhong Park, U. Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5845–5854, 2020. 1, 3

[24] Keunhong Park, U. 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*, abs/2106.13228, 2021. 1, 3

[25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2020. 3

[26] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. Lolnerf: Learn from one look, 2022. 2

[27] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[28] Anju Tewari, Otto Fried, Justus Thies, Vincent Sitzmann, S. Lombardi, Z. Xu, Tanaba Simon, Matthias Nießner, Edgar Tretschk, L. Liu, Ben Mildenhall, Pranatharthi Srinivasan, R. Pandey, Sergio Orts-Escolano, S. Fanello, M. Guang Guo, Gordon Wetzstein, J y Zhu, Christian Theobalt, Manju Agrawala, Donald B. Goldman, and Michael Zollhöfer. Advances in neural rendering. *Computer Graphics Forum*, 41, 2021. 3

[29] Edgar Tretschk, Ayush Kumar Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12939–12950, 2020. 1, 3

[30] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 2

[31] 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

[32] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32653–32666. Curran Associates, Inc., 2022. 1

[33] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9416–9426, 2020. 3

[34] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*, 2022. 2

[35] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5335–5344, 2020. 2, 3, 4, 6

[36] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2