# Temporal Interpolation Is All You Need for Dynamic Neural Radiance Fields

Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, Nahyup Kang

Samsung Advanced Institute of Technology (SAIT)

{sh2019.park, minjungs.son, swan.jang, ychun.ahn, jiyeon31.kim, nahyup.kang}@samsung.com
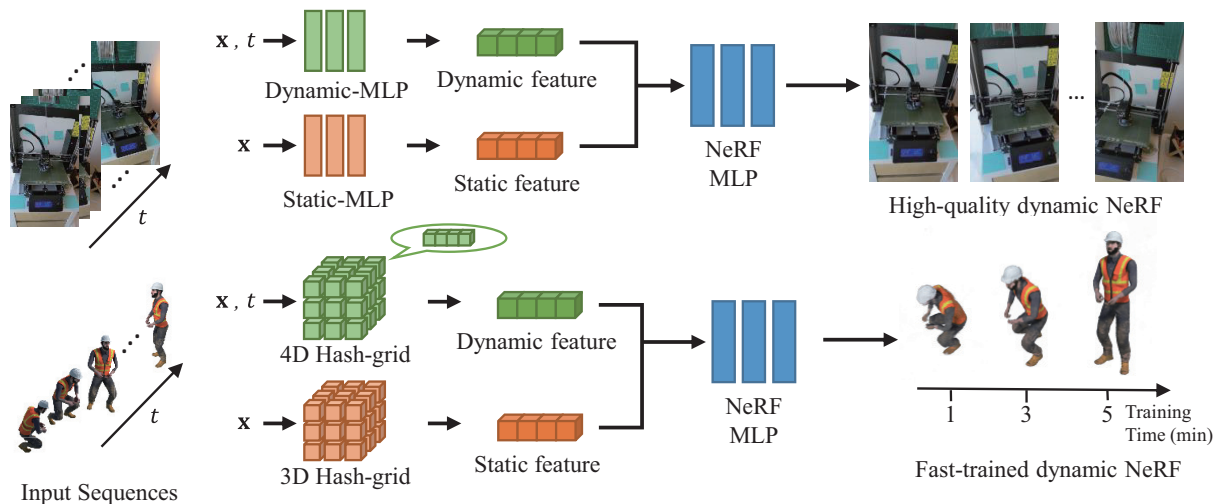
Figure 1. We propose simple yet effective feature interpolation methods for training neural radiance fields of dynamic scenes based on temporal interpolation. We provide two different feature vector representations, neural representation (top) and grid representation (bottom), both of which are the concatenation of static feature vectors and temporally-interpolated dynamic feature vectors. The neural representation exhibits high-quality rendering performance with small-sized models while the grid representation shows competitive rendering results with astonishingly fast training speed.

## Abstract

*Temporal interpolation often plays a crucial role to learn meaningful representations in dynamic scenes. In this paper, we propose a novel method to train spatiotemporal neural radiance fields of dynamic scenes based on temporal interpolation of feature vectors. Two feature interpolation methods are suggested depending on underlying representations, neural networks or grids. In the neural representation, we extract features from space-time inputs via multiple neural network modules and interpolate them based on time frames. The proposed multi-level feature interpolation network effectively captures features of both short-term and long-term time ranges. In the grid representation, space-time features are learned via four-dimensional hash grids, which remarkably reduces training time. The grid representation shows more than 100× faster training speed than the previous neural-net-based methods while maintaining*

*the rendering quality. Concatenating static and dynamic features and adding a simple smoothness term further improve the performance of our proposed models. Despite the simplicity of the model architectures, our method achieved state-of-the-art performance both in rendering quality for the neural representation and in training speed for the grid representation.*

## 1. Introduction

3D reconstruction and photo-realistic rendering have been long-lasting problems in the fields of computer vision and graphics. Along with the advancements of deep learning, differentiable rendering [11, 14] or neural rendering, has emerged to bridge the gap between the two problems. Recently proposed Neural Radiance Field (NeRF) [18] has finally unleashed the era of neural rendering. Using NeRF,

it is able to reconstruct accurate 3D structures from multiple 2D images and to synthesize photo-realistic images from unseen viewpoints. Tiny neural networks are sufficient to save and retrieve complex 3D scenes, which can be trained in a self-supervised manner given 2D images and camera parameters.

Meanwhile, as our world typically involves dynamic changes, it is crucial to reconstruct the captured scene through 4D spatiotemporal space. Since it is often not possible to capture scenes at different viewpoints simultaneously, reconstructing dynamic scenes from images is inherently an under-constrained problem. While NeRF was originally designed to deal with only static scenes, there have been a few approaches in the literature that extend NeRF to dynamic scenes [13, 22, 23, 25], which are so called as dynamic NeRFs. Inspired by the non-rigid structure-from-motion algorithms [1, 4] that reconstruct 3D structures of deformable objects, most previous works solved the under-constrained setting by estimating scene deformations [21, 22, 25] or 3D scene flows [5, 9, 12] for each frame.

However, since the parameters of deformation estimation modules are jointly optimized with NeRF network simultaneously, it is questionable that the modules can accurately estimate deformations or scene flows in accordance with its design principles. In many dynamic scenes, it is challenging to resolve the ambiguities whether a point was newly appeared, moved, or changed its color. It is expected that those ambiguities and deformation estimation can be separately solved within a single network, but in practice, it is hard to let the network implicitly learn the separation, especially for general dynamic scenes without any prior knowledge of deformation.

On the other hand, grid representations in NeRF training [7, 19, 24] have grabbed a lot of attentions mainly due to its fast training speed. Simple trilinear interpolation is enough to fill in the 3D space between grid points. While the representation can be directly adopted to dynamic NeRFs together with the warping estimation module [6], it still requires additional neural networks that affects training and rendering speed.

Motivated by the aforementioned analyses, we present a simple yet effective architecture for training dynamic NeRFs. The key idea in this paper is to apply feature interpolation to the temporal domain instead of using warping functions or 3D scene flows. While the feature interpolation in 2D or 3D space has been thoroughly studied, to the best of our knowledge, feature interpolation method in temporal domain for dynamic NeRF has not been proposed yet. We propose two multi-level feature interpolation methods depending on feature representation which is either neural nets or hash grids [19]. Overview of the two representations, namely the neural representation and the grid representation, are illustrated in Fig. 1. In addition,

noting that 3D shapes deform smoothly over time in dynamic scenes, we additionally introduced a simple smoothness term that encourages feature similarity between the adjacent frames. We let the neural networks or the feature grids to learn meaningful representations implicitly without imposing any constraints or restrictions but the smoothness regularizer, which grants the flexibility to deal with various types of deformations. Extensive experiments on both synthetic and real-world datasets validate the effectiveness of the proposed method. We summarized the main contributions of the proposed method as follows:

- We propose a simple yet effective feature extraction network that interpolates two feature vectors along the temporal axis. The proposed interpolation scheme outperforms existing methods without having a deformation or flow estimation module.

- We integrate temporal interpolation into hash-grid representation [19], which remarkably accelerates training speed more than $100\times$ faster compared to the neural network models.

- We propose a smoothness regularizer which effectively improves the overall performance of dynamic NeRFs.

## 2. Related Work

**Dynamic NeRF.** There have been various attempts to extend NeRF to dynamic scenes. Existing methods can be categorized into three types: warping-based, flow-based, and direct inference from space-time inputs.

Warping-based methods learn how 3D structure of the scene is deformed. 3D radiance field for each frame is warped to single or multiple canonical frames using the estimated deformation. Deformation is parameterized as 3D translation [23], rotation and translation via angle-axis representation [21, 22], or weighted translation [25].

On the other hand, flow-based methods estimate the correspondence of 3d points in consecutive frames. Neural scene flow fields [13] estimates 3D scene flow between the radiance fields of two time stamps. Xian *et al*. [27] suggested irradiance fields which spans over 4D space-time fields with a few constraints derived from prior knowledge.

Although warping-based and flow-based approaches showed successful results, they have a few shortcomings. For instance, warping-based methods cannot deal with topological variations. Since those methods warp every input frame to a single canonical scene, it is hard to correctly represent newly appeared or disappeared 3D radiance fields in the canonical scene. HyperNeRF [22] learns hyperspace which represents multidimensional canonical shape bases. Neural scene flow fields [13] solve the problem by introducing occlusion masks which are used to estimate the regions where scene flows are not applicable. However, the masks

and the flow fields should be learned simultaneously during the training, which makes the training procedure complicated and imposes dependence on extra information such as monocular depth or optical flow estimation.

Without estimating shape deformations or scene flows, DyNeRF [12] used a simple neural network to train NeRFs of dynamic scenes, although the experiments are only conducted on synchronized multi-view videos. It utilizes 3D position and time frame as inputs, and predicts color and density through a series of fully connected neural networks. Our neural representation applies temporal interpolation to intermediate features which enhances representation power for dynamic features while keeping the simple structure.

Category-specific NeRFs are one of the popular research directions for training dynamic objects such as humans [8, 26] or animals [28]. They are based on parametric models or use additional inputs such as pose information. Our method focuses on training NeRFs of general dynamic scenes without any prior knowledge about scenes or objects. Nevertheless, temporal interpolation proposed in this paper can be easily applied to the dynamic NeRFs with templates. **Grid representations in NeRF.** One of the major drawbacks of the original NeRF [18] is slow training speed. A few concurrent works are proposed to speed up NeRF training. Among them, grid-based representation such as Plenoxels [7], direct voxel grid optimization [24] show superior performance in terms of training time, where it takes only a few minutes to learn plausible radiance fields. Combining grid representation with a hash function further improves the efficiency and training speed of the feature grids [19]. Fang *et al.* [6] firstly applied a voxel grid representation to train dynamic NeRFs and achieved much faster training speed compared to the neural-network-based methods. Guo *et al.* [10] also proposed a deformable voxel grid method for fast training. Unlike aforementioned works, our grid representation does not estimate warping or deformation. Instead, it directly estimates color and density using the features obtained from 4D hash grids, which decreases the computational burden and enables faster training.

## 3. Temporal Interpolation for Dynamic NeRF

### 3.1. Preliminaries

Given 3D position $\mathbf{x} \in \mathbb{R}^3$ and 2D viewing direction $\mathbf{d} \in \mathbb{R}^2$, NeRF [18] aims to estimate volume density $\sigma \in \mathbb{R}$ and emitted RGB color $\mathbf{c} \in \mathbb{R}^3$ using neural networks. We formulate dynamic NeRF as a direct extension from 3D to 4D by adding time frame $t \in \mathbb{R}$ to inputs, *i.e.*

$$(\mathbf{c}, \sigma) = f(\mathbf{v}(\mathbf{x}, t), \mathbf{d}). \tag{1}$$

$f$ is implemented as fully connected neural networks and a space-time feature representation $\mathbf{v}$ can be MLP-based neural nets or explicit grid values as explained in Sec. 3.2.

From the camera with origin $\mathbf{o}$ and ray direction $\mathbf{d}$, the color of camera ray $\mathbf{r}(u) = \mathbf{o} + u\mathbf{d}$ at time frame $t$ is

$$\mathbf{C}(\mathbf{r}, t) = \int_{u_n}^{u_f} U(u, t)\sigma(\mathbf{v})\mathbf{c}(\mathbf{v}, \mathbf{d})du, \tag{2}$$

where $u_n$ and $u_f$ denote the bounds of the scene volume, $\mathbf{v}$ is an abbreviation of $\mathbf{v}(\mathbf{r}(u), t)$ and $U(u, t) = \exp(-\int_{u_n}^{u} \sigma(\mathbf{v}(\mathbf{r}(s), t))ds)$ is the accumulated transmittance from $u_n$ to $u$. Then, the RGB color loss $L_c$ is defined to minimize the $l_2$-loss between the estimated colors $\hat{C}(\mathbf{r}, t)$ and the ground truth colors $C(\mathbf{r}, t)$ over all rays $\mathbf{r}$ in camera views $R$ and time frames $t \in T$ as follows:

$$L_c = \sum_{\mathbf{r} \in R, t \in T} ||\hat{C}(\mathbf{r}, t) - C(\mathbf{r}, t)||_2^2. \tag{3}$$

### 3.2. Space-Time Feature Representation

The main contribution of our paper lies on the novel feature representation method for dynamic NeRFs. We define the feature vector $\mathbf{v}$, which is fed into the neural network to determine a color and a volume density, as the concatenation of static and dynamic feature vectors, *i.e.*,

$$\mathbf{v}(\mathbf{x}, t) = [\mathbf{v}_s(\mathbf{x}), \mathbf{v}_d(\mathbf{x}, t)], \tag{4}$$

where $[\cdot, \cdot]$ means concatenation operator of two vectors. The static feature vector $\mathbf{v}_s$ only depends on the 3D position $\mathbf{x}$. As most dynamic scenes also contain static regions such as background, $\mathbf{v}_s$ is designed to learn the static features that are consistent across time. Meanwhile, $\mathbf{v}_d$ learns dynamic features which may vary across time. We propose two novel feature representation methods in which temporal interpolation is applied. First, the neural representation, which is essentially a series of neural networks combined with linear interpolation, is suggested in Sec. 3.2.1. The model trained using the neural representation is able to render high-quality space-time view synthesis with small-sized neural networks ($\sim$20MB). Second, the grid representation, which is an temporal extension of recently proposed voxel grid representations [19], is explained in Sec. 3.2.2. Dynamic NeRFs can be trained in less than 5 minutes with the proposed grid representation.

#### 3.2.1 Neural Representation

In the neural representation, features that fed into the template NeRFs are determined by a series of neural nets. In other words, both static and dynamic feature extractor are formulated as multi-layer perceptrons (MLPs). First, the whole time frame is divided into equally spaced time slots. For each time slot, two MLPs are assigned. Then the whole feature vector is interpolated from the assigned MLPs. Overall feature extraction procedure is illustrated in Fig. 2.
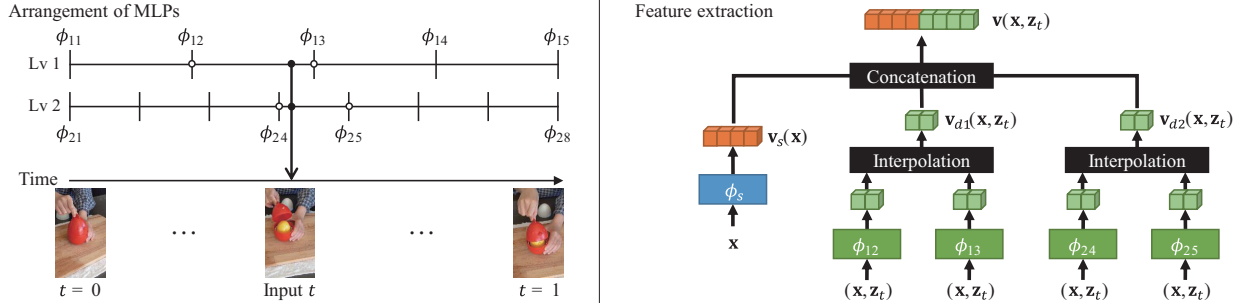
Figure 2. Illustration of the arrangements of MLPs (left) and the feature extraction (right) in the neural representation. $\phi_s$ and $\phi_{ij}$ are small MLPs that extract static and dynamic features respectively. The static features $\mathbf{v}_s$ and the dynamic feature $\mathbf{v}_d$ are concatenated and fed into the template NeRF. Detailed structures of the template NeRF can be found in the supplementary materials.

Concretely, let $\mathbf{v}_d(\mathbf{x}, \mathbf{z}_t)$ be the feature vector for a 3D point $\mathbf{x}$ at time $t$, which will be fed to the template NeRF. Here, we used $\mathbf{z}_t$ as an embedding vector for an input time frame $t$ [12, 16, 22]. For equally spaced $n$ time slots, there are $n+1$ keyframes $t_i = \frac{i}{n}(i = 0, 1, 2, \ldots, n)$. An MLP $\phi_i$ is assigned to each keyframe $t_i$ which is responsible for two adjacent time slots $[t_{i-1}, t_i]$ and $[t_i, t_{i+1}]$. For inputs with time $t \in [t_i, t_{i+1}]$, $\mathbf{x}$ and $\mathbf{z}_t$ are fed into $\phi_i$ and $\phi_{i+1}$. Then, the outputs from two MLPs are interpolated as follows:

$$\mathbf{v}_d(\mathbf{x}, \mathbf{z}_t) = \Delta t \cdot \phi_i(\mathbf{x}, \mathbf{z}_t) + (1 - \Delta t) \cdot \phi_{i+1}(\mathbf{x}, \mathbf{z}_t) \quad (5)$$

where $\Delta t = (t_{i+1} - t)/(t_{i+1} - t_i)$.

The purpose of this interpolation is to efficiently learn the features between keyframes in a scalable manner. Thanks to the temporal interpolation, we can learn the features of the continuous time range $[t_i, t_{i+1}]$ by enforcing the MLPs $\phi_i$ and $\phi_{i+1}$ responsible for that time range. While the static feature $\mathbf{v}_s$ represents the features across the whole timeline, $\mathbf{v}_d$ represents the features that are focused more on dynamic regions. In addition, since each MLP for dynamic feature is responsible only for two adjacent time slots, it is able to make each $\phi_i$ learn features that are specific to a certain period of time.

To exploit the features of both long-term and short-term, there can be multiple levels of dynamic features which have different number of keyframes. For multi-level dynamic feature extractor with level $l$, each level contains different number of keyframes $n_1, n_2, \cdots, n_l$. Let $\mathbf{v}_{di}$ denote the dynamic feature of $i$-th level, then the output dynamic feature is the concatenation of features from all levels, *i.e.*,

$$\mathbf{v}_d(\mathbf{x}, \mathbf{z}_t) = [\mathbf{v}_{d1}(\mathbf{x}, \mathbf{z}_t), \mathbf{v}_{d2}(\mathbf{x}, \mathbf{z}_t), \cdots, \mathbf{v}_{dl}(\mathbf{x}, \mathbf{z}_t)]. \quad (6)$$

In this paper, we used the settings $l = 2, n_1 = 5, n_2 = 20$ otherwise stated. The dimensions of feature vectors are determined as 128 for $\mathbf{v}_s$, and 64 for $\mathbf{v}_0$ and $\mathbf{v}_1$. MLP with one hidden layer whose hidden size is the same as its output dimension is used for feature extraction of both $\mathbf{v}_s$ and $\mathbf{v}_{di}$.

The concatenated feature vectors are fed to the template NeRF which outputs volume density $\sigma$ and emitted color $\mathbf{c}$. We used the same structure as used in HyperNeRF [22] for the template NeRF of the neural representation.

### 3.2.2 Grid Representation

Recently, InstantNGP [19] suggested a novel multi-level grid representation with hash tables. We adopt the hash grid representation from [19] and extend it for fast dynamic NeRF training. Similar to the neural representation in Sec. 3.2.1, the feature vector from the proposed hash grid contains static and dynamic feature vectors. The static feature $\mathbf{v}_s$ is analogous to the one used in [19]. On the other hand, the dynamic feature $\mathbf{v}_d$ comes from the 4D hash grids.

Concretely, to extract static and dynamic feature vectors whose dimensions are $m_s$ and $m_d$ respectively, a hash table of size $H$ that contains $(m_s + m_d)$ dimension feature vectors is constructed. The hash function of $d$-dimensional vector, $h_d(\mathbf{x})$, is defined as

$$h_d(\mathbf{x}) = \bigoplus_{i=1}^{d} (x_i P_i) \mod H, \quad (7)$$

where $P_i$ is a large prime number and $\bigoplus$ is an XOR operator. Then, the feature vector $\mathbf{v}$ is retrieved by concatenating the outputs of 3D and 4D hash functions:

$$\mathbf{v}(\mathbf{x}, t) = [h_3(\mathbf{x}), h_4(\mathbf{x}, t)]. \quad (8)$$

The 3D and 4D grids are constructed as the multi-level hash grids proposed in [19]. We applied different scaling factors for 3D space and time frame since the number of frames for training sequences are usually much smaller than the finest 3D grid resolutions.

### 3.3. Smoothness Regularization

As our dynamic world smoothly changes, it is reasonable to impose a smoothness term to adjacent time frames.
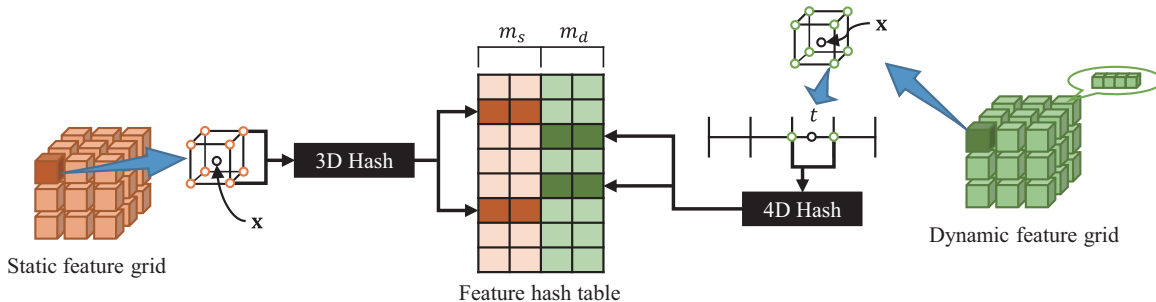
Figure 3. Illustration of grid representation with hash tables at a certain level. a static feature vector of a 3D grid point is extracted via 3D hash function, while dynamic feature vector for 4D grid point is extracted via 4D hash. The static feature vector $\mathbf{v}_s$ is determined as trilinear interpolation of 8 vectors of grid points, while the dynamic feature vector $\mathbf{v}_d$ is calculated as quadrilinear interpolation of 16 vectors. In the figure, we only depicted feature retrieval process of two grid points for illustration purpose.

We note that the smoothness term is only applied to the input feature space, not to the estimated outputs such as RGB color or density. For the neural representation, we provided a simple smoothness term which is calculated as,

$$L_s(\mathbf{x}, t) = \|\mathbf{v}_d(\mathbf{x}, \mathbf{z}_t) - \mathbf{v}_d(\mathbf{x}, \mathbf{z}_{t+1})\|_2^2. \qquad (9)$$

This regularization term has two advantages. First, it reflects the intuition that the observed point $\mathbf{x}$ at time $t$ will be stationary if there is no observation for $\mathbf{x}$ at time $t + 1$. Second, by propagating additional gradients to the feature networks or the hash grids, the smoothness term acts as a regularizer that stabilizes the training.

For the grid representation, we impose a smoothness term to the grid points that are temporally adjacent:

$$L_s(\mathbf{x}, t) = \frac{1}{n_f^2} \|h_4(\mathbf{x}, t_a) - h_4(\mathbf{x}, t_b)\|_2^2, \qquad (10)$$

where $n_f$ is the number of frames in the training sequence and $t_a, t_b$ are two adjacent grid points that satisfies $t_a \leq t \leq t_b$. In fact, Eq. (10) can be obtained from Eq. (9) in the grid representation, which indicates that imposing smoothness term to adjacent time frames is equivalent to add smoothness to two temporally adjacent grids with constant multiplier. Detailed derivation of this relationship is provided in the supplementary materials.

The smoothness term is added to the loss function with a weight $\lambda$, and the total loss is minimized together during the training. Accordingly, the total loss of our dynamic neural radiance fields is given by

$$L = L_c + \lambda L_s. \qquad (11)$$

We observe that the smoothness term is especially powerful in the static background regions that appeared only in a small fraction of frames. In Fig. 4, we show examples of the rendered images which are trained with and without the smoothness term. It can be clearly seen that the boxed



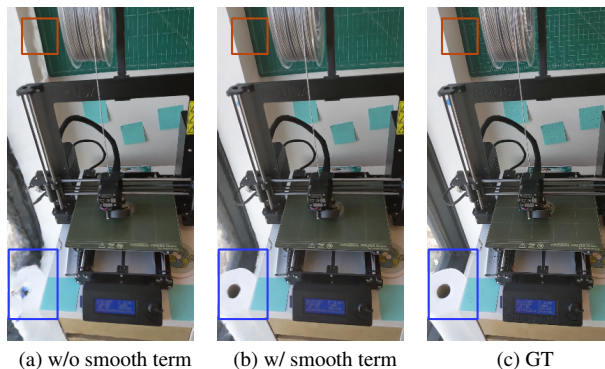(a) w/o smooth term  (b) w/ smooth term  (c) GT

Figure 4. Effectiveness of the smoothness term. It can be observed that the model trained with the smoothness term accurately renders the corner part which appears in only few frames within the whole sequence (blue box) and removes spurious artifacts (red box).

regions of the rendered image shows much plausible when the model is trained with the smoothness term.

### 3.4. Implementation Details

In the neural representation, we adopted the template NeRF architecture of HyperNeRF [22] for density and color estimation. The network is an 8-layered MLP with hidden size of 256. The dimension of the embedding vector $\mathbf{z}_t$ is set to 8. The smoothness weight $\lambda$ is set to 0.01 or 0.001 depending on the characteristics of the datasets used in the experiments. We set $\lambda$ to a large value in the sequences that the viewpoint change is significant in a short period of time although performance variations depending on the value of $\lambda$ is not significant.

In the grid representation, $\lambda$ is set to $1e^{-4}$. We applied the smoothness loss only for the finest two levels of the temporal dimension since applying it to every level slows training speed without much improvements in performance. After feature vectors are extracted from the hash grids, we fed them to a 3-layered MLP with hidden size of 128 to estimate

volume density followed by one additional layer for RGB color estimation similar to [19]. We set $m_s = 2, m_d = 6$ and $H = 2^{19}$ for the grid representation experiments otherwise stated. The 3D and 4D hash grids are composed of 12 levels. For the spatial dimension, we set the base resolution to 8, and the scaling factor is set to 1.45. The base resolution of the temporal dimension is 2, which is multiplied by 1.4 in every other level. Detailed hyperparameter settings and network architectures can be found in the supplementary materials.

## 4. Experimental Results

### 4.1. Datasets

To validate the superiority of our method, we conducted extensive experiments on various datasets. We used three publicly available datasets that are used for dynamic NeRFs. For all experiments, we trained our models for each sequence individually, and then per-scene results are averaged and reported in this section.

**D-NeRF Dataset [23].** The dataset consists of synthetically rendered images of moving and deforming 3D objects. For each frame, the image is rendered via a synthetic camera of random rotation. There are eight scenes in total, and each scene contains 50-200 training images with 20 test views.

**HyperNeRF Dataset [22].** The dataset contains video sequences taken from mobile phones. There are four sequences in *vrig-dataset* which are taken using the camera rig with two phones vertically aligned. In addition, there are six sequences in *interp-dataset* which are taken from a single camera in order to estimate the image in the middle of two consecutive training images.

**DyNeRF Dataset [12].** The dataset consists of videos obtained from a capture system that consists of 21 GoPro Black Hero cameras. The cameras are located at a fixed position, and all video frames are synchronized to build the multi-view video dataset.

### 4.2. Neural Representation

In this section, we reported the performance of the proposed neural representation models. First, the experimental results on D-NeRF dataset are shown in Tab. 1. Peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and perceptual similarity (LPIPS) [29] are used as evaluation metrics following the previous works. We also reported the average metric (AVG) proposed in [3] which aggregates three metrics to a single value. Our method with the neural representation (Ours-NN) achieves state-of-the-art results on all evaluation metrics. It is worth noting that the smoothness term dramatically improves overall performance.

Next, we evaluated our method on HyperNeRF dataset [22] and compared with existing methods, which is shown in Tab. 2. Here, we reported PSNR and multi-

| | PSNR↑ | SSIM↑ | LPIPS↓ | AVG↓ |
|---|---|---|---|---|
| NeRF [18] | 19.00 | 0.87 | 0.18 | 0.09 |
| T-NeRF [23] | 29.50 | 0.95 | 0.08 | 0.03 |
| D-NeRF [23] | 30.43 | 0.95 | 0.07 | 0.02 |
| NDVG-full [10] | 27.84 | 0.862 | 0.041 | 0.029 |
| TiNeuVox-B [6] | 32.67 | 0.971 | 0.041 | 0.016 |
| Ours-NN (w/o smooth) | 30.18 | 0.963 | 0.038 | 0.019 |
| Ours-NN (w/ smooth) | **32.73** | **0.974** | **0.033** | **0.014** |

Table 1. Experimental results on D-NeRF [23] datasets.

scale SSIM (MS-SSIM), and excluded LPIPS metric since its value cannot reliably reproduced [6]. Our method shows second-best results on both *vrig* and *interp* datasets. While flow-based method [13] suffers from interpolating motions between two consecutive frames, our method, which implicitly learns intermediate feature representation of in-between frames, achieves competitive performance with warping-based methods [21, 22].

Qualitative results for the neural representation are shown in Fig. 5. It is clearly observed that our method captures fine details compared to D-NeRF [23]. We highlighted the regions that show notable differences with colored boxes and placed zoomed-in images of the regions next to the rendered images. In addition, we also qualitatively compared our method with HyperNeRF [22] on their datasets. When the warping estimation module of [22] does not correctly estimate warping parameters, HyperNeRF produces implausible results. It can be observed that the head and the body of the chicken toy is not properly interlocked and the position of the hand peeling banana is incorrect. On the other hand, our method accurately recovers 3D geometry of those scenes. Thus, without using the physically meaningful warping estimation module in neural networks, the proposed temporal interpolation and the smoothness regularizer provides simple yet effective way to learn complex deformations.

Lastly, performance evaluation on DyNeRF dataset [12] is presented in Tab. 3. We adopt PSNR, LPIPS, and FLIP [2] for evaluation metrics to compare with previous works. Our method achieves best PSNR while ranked second in LPIPS and FLIP. However, those two metrics are also better than DyNeRF[†] [12] which does not use importance sampling strategy in [12]. Since we do not use any sampling strategy during training, it can be concluded that our feature interpolation method is superior to the network architecture of [12]. Notably, our method outperforms DyNeRF with smaller network size (20MB) than the DyNeRF models (28MB).

### 4.3. Grid Representation

We used D-NeRF datasets to evaluate the performance of the proposed grid representation. Since this representation is mainly intended for fast training, we report the re-
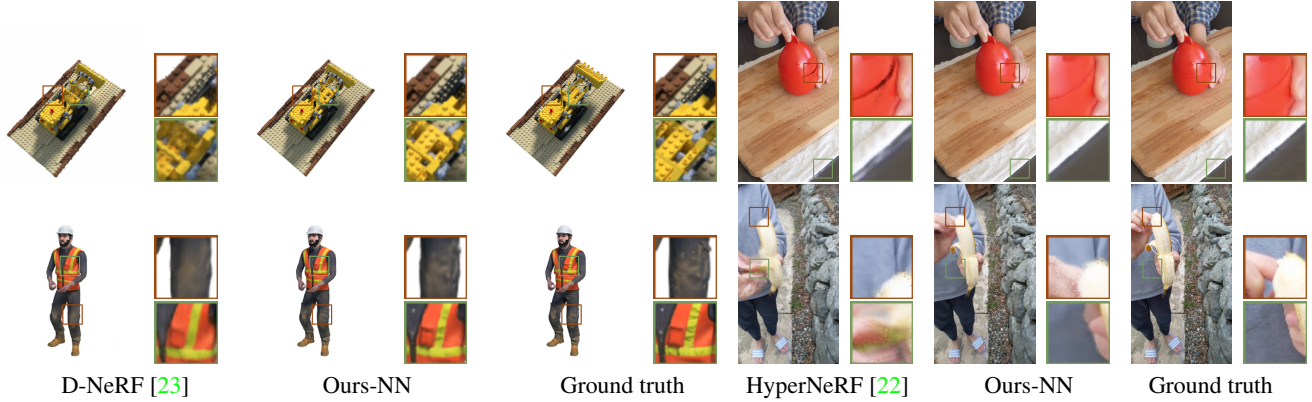
Figure 5. Qualitative results of our method (Ours-NN) on of D-NeRF [23] and HyperNeRF [22] datasets. The regions that show significant difference with compared methods are scaled up (red and green boxes).

|  | vrig | | interp | |
|---|---|---|---|---|
|  | PSNR↑ | MS-SSIM↑ | PSNR↑ | MS-SSIM↑ |
| NeRF [18] | 20.13 | 0.745 | 22.27 | 0.804 |
| NV [15] | 16.85 | 0.571 | 26.05 | 0.911 |
| NSFF [13] | **26.33** | **0.916** | 25.80 | 0.883 |
| Nerfies [21] | 22.23 | 0.803 | 28.47 | 0.939 |
| HyperNeRF [22] | 22.38 | 0.814 | **29.00** | **0.945** |
| Ours-NN | <u>24.35</u> | <u>0.867</u> | <u>28.67</u> | <u>0.940</u> |

Table 2. Experimental results on HyperNeRF [22] datasets.

|  | PSNR↑ | LPIPS↓ | FLIP↓ |
|---|---|---|---|
| MVS | 19.12 | 0.2599 | 0.2542 |
| NeuralVolumes [15] | 22.80 | 0.2951 | 0.2049 |
| LLFF [17] | 23.24 | 0.2346 | 0.1867 |
| NeRF-T [12] | 28.45 | 0.100 | 0.1415 |
| DyNeRF† [12] | 28.50 | 0.0985 | 0.1455 |
| DyNeRF [12] | <u>29.58</u> | **0.0832** | **0.1347** |
| Ours-NN | **29.88** | <u>0.0960</u> | <u>0.1413</u> |

Table 3. Experimental results on DyNeRF [12] datasets.

|  | Train time | PSNR | SSIM | LPIPS | AVG |
|---|---|---|---|---|---|
| InstantNGP [19] | 5 min | 20.28 | 0.888 | 0.146 | 0.077 |
| D-NeRF [23] | 20 hours | 30.43 | 0.95 | 0.07 | 0.02 |
| NDVG-full [10] | 35 min | 27.84 | 0.862 | 0.041 | 0.029 |
| TinueVox-B [6] | 30 min | 32.67 | 0.971 | 0.041 | 0.016 |
| NDVG-half [10] | 23 min | 27.15 | 0.857 | 0.048 | 0.033 |
| TinueVox-S [6] | 8 min | 30.75 | 0.956 | 0.067 | 0.023 |
| Ours-grid | 1 min | 26.77 | 0.933 | 0.107 | 0.039 |
| Ours-grid | 5 min | 29.73 | 0.961 | 0.063 | 0.024 |
| Ours-grid | 8 min | 29.84 | 0.962 | 0.062 | 0.023 |

Table 4. Quantitative comparison of training time and performance on D-NeRF datasets using the grid representation.

|  | D-NeRF dataset | | | HyperNeRF vrig | |
|---|---|---|---|---|---|
|  | PSNR | SSIM | LPIPS | PSNR | MS-SSIM |
| NN (dynamic only) | 30.91 | 0.963 | 0.043 | 24.07 | 0.861 |
| NN (dynamic+static) | **32.73** | **0.974** | **0.033** | 24.35 | **0.866** |
| Grid (dynamic only) | 29.08 | 0.948 | 0.076 | 22.11 | 0.755 |
| Grid (dynamic+static) | **29.84** | **0.962** | **0.062** | 22.98 | **0.802** |

Table 5. Ablation study on the effectiveness of static features.

sults in a short period time (∼8 minutes) and compare the results with the concurrent works [6, 10], both of which are based on voxel grid representation and showed the fastest training speed for dynamic NeRFs so far. We also examined the performance of the original implementation of InstantNGP [19] as a baseline with no temporal extension. All of the grid representation models in the experiments are trained on a single RTX 3090 GPU for fair comparison with [6, 10].

The quantitative results are shown with elapsed training time in Tab. 4. The grid representation demonstrates much faster training speed even compared to the recent voxel-grid based methods [6, 10]. In comparison with neural network models [23], our method exceeds SSIM and LPIPS of [23] in 5 minutes which have arithmetically 240 times faster training speed. By taking the benefits of fully fused neural network [20] as well as the efficient hash grid repre-

sentation, our method quickly learns meaningful features in dynamic scenes and generates acceptable novel view synthesis results with just one minute of training. Our method shows superior SSIM, LPIPS and slightly inferior PSNR to TinueVox-S [6], which indicates clearer and more detailed rendering results, when trained for the same period of time.

We depict training progress of the grid representation qualitatively in Fig. 6. We also compared the results of TinueVox-S [6] which are trained for 8 minutes. After one minute of training, the grid representation model produces blurry images but accurately render dynamic regions compared to InstantNGP [19]. After 5 minutes, the model successfully renders sharp images that are similar to the ground truth images. Notably, our model exhibits more sharp and clear results than TinueVox-S despite shorter training time, which results in better SSIM and LPIPS..

Finally, we compared the performance of the grid repre-

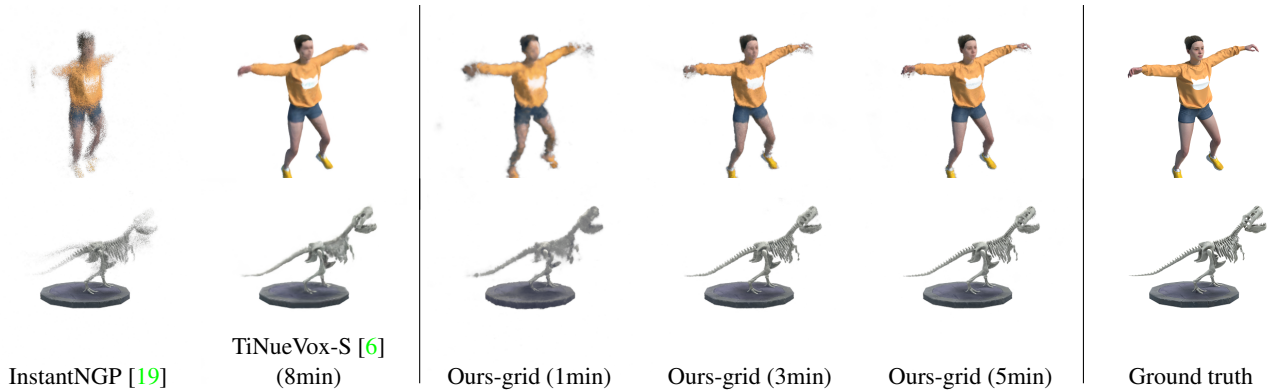| InstantNGP [19] | TiNueVox-S [6] (8min) | Ours-grid (1min) | Ours-grid (3min) | Ours-grid (5min) | Ground truth |

Figure 6. Qualitative results of the grid representation. While InstantNGP [19] is not able to reconstruct dynamic regions, our method produces acceptable rendering results in one minute and performs superior to TiNueVox-S [6] with shorter training time.
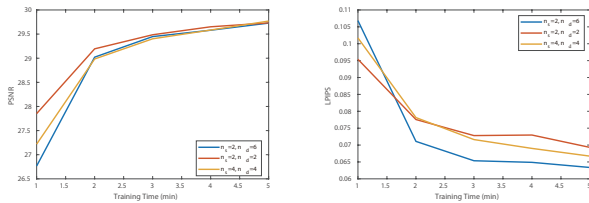


Figure 7. Performance comparison with different feature dimension settings. Using smaller feature dimensions results in faster training but quicker saturation of model performance.



| Ours-NN | Ground truth | Ours-grid | Ground truth |

Figure 8. Examples of failure cases.

sentation by varying the dimension of static and dynamic feature vectors, $m_s$ and $m_d$. PSNR and LPIPS on test images are measured per minute and illustrated in Fig. 7. We trained the grid representation models in three different settings, $m_s = 2, m_d = 6$, $m_s = 2, m_d = 2$, $m_s = 4, m_d = 4$. When smaller feature dimension is used, the training speed is faster, so PSNR increases fast in the early stage of training. However, the performance also saturates faster, and LPIPS is inferior to the other settings. The model with $m_s = 2, m_d = 6$ significantly outperformed the others in terms of LPIPS.

### 4.4. Effectiveness of the Static Features

To validate the effectiveness of the static features, we conducted an ablation study on the static features and illustrated the results in Tab. 5. For both the neural representation (NN) and the grid representation (Grid), the models with static features performs superior to the ones using only dynamic features in all metrics. Additional ablation studies and qualitative results can be found in the supplementary materials.

### 4.5. Failure Cases

Although the proposed feature interpolation is able to learn meaningful spatiotemporal features in most cases,
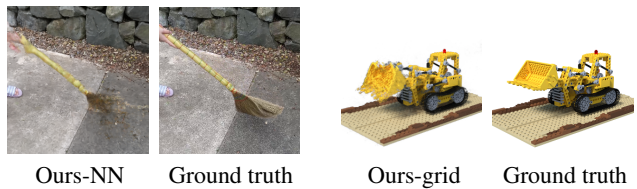
there are a few failure cases as presented in Fig. 8. For instance, our method has difficulty in recovering 3D structures when small objects in a sequence rapidly move (Fig. 8 left) or when there exist dynamic regions that are not observed in the training sequence (Fig. 8 right).

## 5. Conclusion

In this paper, we propose a simple yet effective feature interpolation method for training dynamic NeRFs. Both the neural representation and the grid representation showed impressive performance, and the smoothness term applied to the intermediate feature vectors further improves the performance. Since these methods are unrelated to the existing methods of modeling deformations or estimating scene flows, we believe that the proposed method suggests a new direction of training dynamic NeRFs.

While the neural representation model shows high-quality rendering results owing to the representation power of neural networks, it requires hours of training and seconds of rendering which impose barriers to real-time applications. On the other hand, the grid representation is able to render dynamic scenes in less than a second after a few minutes of training, which makes it more practical for real-world applications. Both representations are mutually complementary, and investigating hybrid representations that take advantages of both representations would be an interesting research direction.

# References

[1] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1442–1456, 2010. 2

[2] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):15:1–15:23, 2020. 6

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 6

[4] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 690–696. IEEE, 2000. 2

[5] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021. 2

[6] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arXiv preprint arXiv:2205.15285*, 2022. 2, 3, 6, 7, 8

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2, 3

[8] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, 2021. 3

[9] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[10] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2022. 3, 6, 7

[11] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 1

[12] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 2, 3, 4, 6, 7

[13] 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, 6, 7

[14] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 1

[15] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 7

[16] 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. 4

[17] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 7

[18] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 3, 6, 7

[19] 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, 3, 4, 6, 7, 8

[20] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. 7

[21] 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, 6, 7

[22] 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, 4, 5, 6, 7

[23] 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, 6, 7

[24] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields

reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 2, 3

[25] Edgar Tretschk, Ayush 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 2

[26] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16210–16220, June 2022. 3

[27] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2

[28] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022. 3

[29] 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