# Sync-NeRF: Generalizing Dynamic NeRFs to Unsynchronized Videos

**Seoha Kim**[1*], **Jeongmin Bae**[1*], **Youngsik Yun**[1],
**Hahyun Lee**[2], **Gun Bang**[2], **Youngjung Uh**[1†]
[1]Yonsei University
[2]Electronics and Telecommunications Research Institute
{hailey07, jaymin.bae, bbangsik, yj.uh}@yonsei.ac.kr, {hanilee, gbang}@etri.re.kr

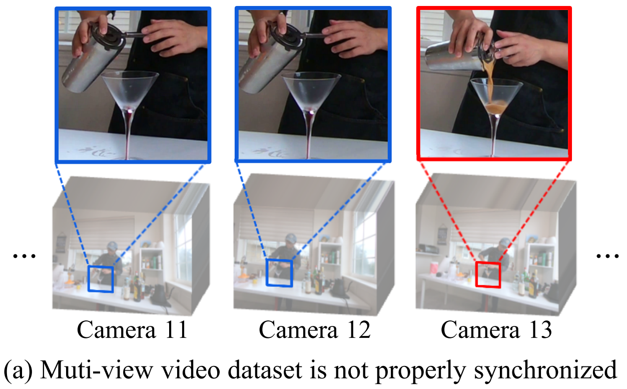
## Abstract

Recent advancements in 4D scene reconstruction using neural radiance fields (NeRF) have demonstrated the ability to represent dynamic scenes from multi-view videos. However, they fail to reconstruct the dynamic scenes and struggle to fit even the training views in *unsynchronized* settings. It happens because they employ a single latent embedding for a frame while the multi-view images at the same frame were actually captured at different moments. To address this limitation, we introduce time offsets for individual unsynchronized videos and jointly optimize the offsets with NeRF. By design, our method is applicable for various baselines and improves them with large margins. Furthermore, finding the offsets naturally works as synchronizing the videos without manual effort. Experiments are conducted on the common Plenoptic Video Dataset and a newly built Unsynchronized Dynamic Blender Dataset to verify the performance of our method. Project page: https://seoha-kim.github.io/sync-nerf

## Introduction

Neural radiance fields (NeRF, Mildenhall et al. 2021) aim to synthesize novel views of a scene when given its limited number of views. As NeRF is a function that receives 3D coordinates with viewing directions and produces color and density, adding time as input inherently generalizes it to design dynamic NeRFs for multi-view videos. Recent works have improved efficiency (Li et al. 2022b; Fang et al. 2022; Wang et al. 2022b,a; Fridovich-Keil et al. 2023) and streamability (Li et al. 2022a; Song et al. 2023; Attal et al. 2023) of dynamic NeRFs.

Our research is motivated by the inaccurate video synchronization in a widely used multi-view dynamic dataset (Li et al. 2022b). Although typical synchronization approaches, such as timecode systems or audio peaks, usually work, these processes require an additional device, cannot be applied in noisy environments, or can be inaccurate (Li et al. 2022b). For example, Figure 1a shows that the rightmost video is severely ahead of others in the temporal axis. While previous dynamic NeRFs show high-fidelity reconstruction by omitting the unsynchronized video, they fail to reconstruct this video even in the training view if it is included (Figure 1b). If we perturb the synchronization of the multi-view videos on purpose, all methods fail to reconstruct movements and produce severe artifacts and ghost effects.



(a) Muti-view video dataset is not properly synchronized

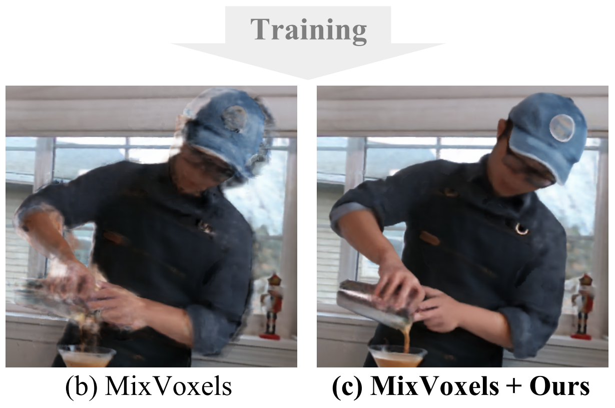


Figure 1: Overview. (a) The commonly used Plenoptic Video Dataset in 4D scene reconstruction contains an unsynchronized video. Image patches are all first frames. (b) If we include this view in the training set, baselines fail to reconstruct the motion around the unsynchronized viewpoint. (c) In the same settings, our method significantly outperforms.

The above drawbacks occur because existing dynamic NeRFs assume that the same frames in multi-view videos are captured at the same time (Figure 2a), which is not always true. Even if the videos are assumed to be synchro-

[*]These authors contributed equally.
[†]Corresponding author.

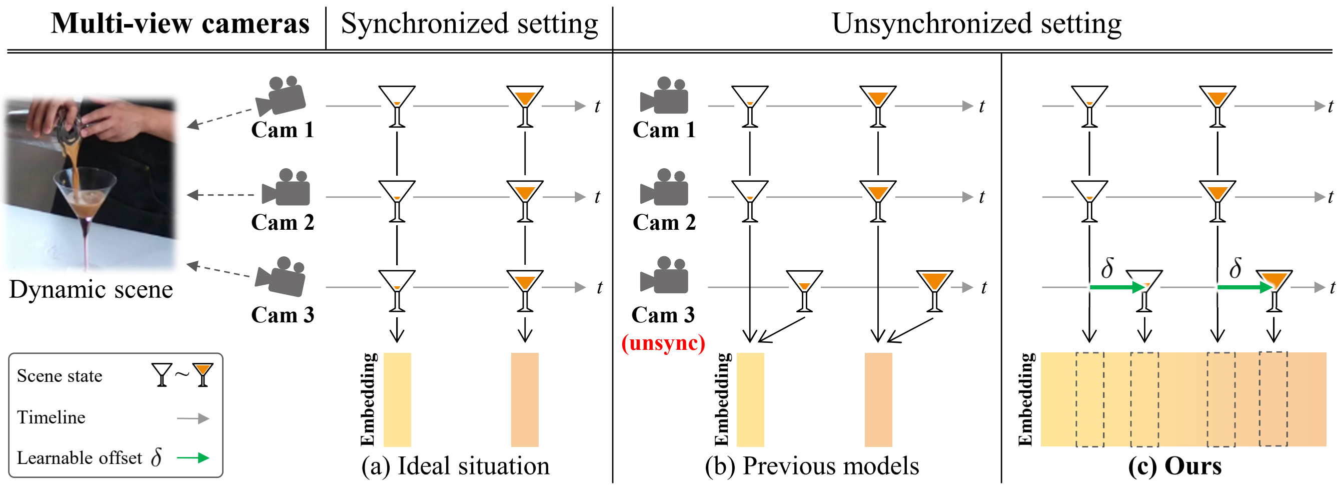


Figure 2: Problem statement. (a) Ideally, all multi-view images at a frame captures the same moment of a scene. Each frame is represented by a latent embedding. (b) Some frames are not synchronized. Previous methods suffer from the discrepancy between the latent embedding of the frame and the actual status of the scene. (c) Our method allows assigning correct temporal latent embeddings to videos captured with temporal gaps by introducing learnable time offsets $\delta$ for individual cameras.

nized, there can be temporal mismatch within a frame. See Appendix A.2 for details. Figure 2b illustrates the common problem: using the same temporal representation for different states of the scene. This problem worsens as the motions in the scene are faster, or as the deviation of time increases. In this paper, we introduce time offsets for individual videos and optimize them jointly with NeRF. It resolves the temporal gap between the observation and the target state caused by unsynchronization (Figure 2c). Consequently, all training videos can be accurately reconstructed as shown in (Figure 1c). As optimizing the time offsets is equivalent to synchronizing the videos by shifting, we name our approach Sync-NeRF. We further design a continuous function that receives time and produces temporal representation to apply our method on dynamic NeRFs with discrete temporal representation (Wang et al. 2022a). We inherit bilinear interpolation for dynamic NeRFs with spatiotemporal planes (Fridovich-Keil et al. 2023).

In order to show multiple advantages of Sync-NeRF, we design new benchmark datasets by randomly unsynchronizing existing datasets and building unsynchronized synthetic scenes with ground truth time offsets.

## Related Work

**Dynamic NeRFs** have been evolving by introducing better representation for specific purposes. D-NeRF or Human-NeRF models deformation field to extend the static NeRF to dynamic domain (Pumarola et al. 2020; Zhao et al. 2022). DyNeRF achieves complex temporal representation by implicitly assigning a latent vector to each frame (Li et al. 2022b). NeRFPlayer or MixVoxels improve the streamability of dynamic scenes by utilizing grid-based NeRF (Li et al. 2022b; Wang et al. 2022a). K-Planes and HexPlane adopt planar factorization for extending to arbitrary dimensions, enabling the representation of dynamic scenes (Fridovich-Keil et al. 2023; Cao and Johnson 2023). These works assume multiple synchronized video inputs but are limited to accurately synchronized multi-view videos. We tackle this limitation by introducing easily optimizable time offsets to the existing methods. Also, our approach can be seamlessly adapted to existing methods.

On the other hand, the methods for monocular video settings are relatively free from the synchronization (Park et al. 2021b; Li et al. 2023). Hence, We do not consider a monocular video setting. Nevertheless, we note that they rely on unnatural teleporting cameras (Gao et al. 2022).

**Per-Frame Latent Embedding** Some methods extend NeRF with multiple latent embeddings to represent multiple scenes or different states of a scene. NeRF-W (Martin-Brualla et al. 2021) and Block-NeRF (Tancik et al. 2022) use per-image appearance embeddings for different states of a scene to reconstruct a scene from unstructured image collections with appearance variations. In dynamic NeRFs, D-NeRF represents dynamic scenes using per-frame deformation embedding. Nerfies and HyperNeRF (Park et al. 2021a,b) apply both per-frame appearance embedding and per-frame deformation embedding. However, per-frame latent embedding approaches cannot represent a dynamic scene from unsynchronized multi-view videos. This is because they share a single latent embedding for multi-view images with the same frame index, even though these images are captured at different moments. Using our time offset method, existing dynamic NeRFs can represent dynamic scenes successfully in unsynchronized settings.

**Joint Camera Calibration** NeRF−−, BARF, and SCNeRF (Jeong et al. 2021; Lin et al. 2021; Jeong et al. 2021) optimize camera parameters and NeRF parameters jointly to eliminate the requirements of known camera parameters in the static setting. RoDyNeRF (Liu et al. 2023) optimizes dynamic NeRF jointly with camera parameters to tackle the failure of COLMAP in highly dynamic scenes. However, previous methods can not compensate for the inaccurate synchronization across multi-view videos. Traditional approaches for synchronization utilize timecode systems (Li et al. 2022b) or audio peaks recorded simultaneously with videos (Shrstha, Barbieri, and Weda 2007). Consequently, these methods require additional devices, and they may not produce accurate results or cannot be applied to videos with significant noise. We overcome these limitations by jointly training per-camera time offsets on top of the existing dynamic NeRFs.

## Method

In this section, we first introduce per-camera time offsets, which enable training dynamic NeRFs on the unsynchronized multi-view videos. Subsequently, we describe an implicit function-based approach for models with per-frame temporal embeddings and an interpolation-based approach for grid-based models.

### Per-Camera Time Offset with Dynamic NeRF

NeRF learns to map a given 3D coordinates $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^3$ to RGB color $\mathbf{c} \in \mathbb{R}^3$ and volume density $\sigma \in \mathbb{R}$. To represent a dynamic scene with NeRF, it is common to modify NeRF as a time-dependent function by adding a time input $t \in \mathbb{R}$:

$$\mathcal{F}_\Theta : (\mathbf{x}, \mathbf{d}, t) \rightarrow (\mathbf{c}, \sigma), \tag{1}$$

where $\Theta$ parameterizes $\mathcal{F}$. Dynamic NeRFs typically employ the video frame index as $t$. Then the model is trained to reconstruct multi-view videos by rendering each frame.

However when the multi-view videos are not synchronized, a single frame index may capture different moments of the scene across the different videos. As a result, the ground truth RGB images captured from different viewpoints at frame $t$ do not match each other, leading to suboptimal reconstruction of dynamic parts.

To this end, we introduce a learnable time offset $\delta_k$ for each of the $K$ training cameras: $t_k = t + \delta_k$. It allows the temporal axis of each video to be freely translated, rectifying potential temporal discrepancies across multi-view videos. The time-dependent function $\mathcal{F}_\Theta$ in Eq. 1 changes accordingly:

$$\mathcal{F}_\Theta : (\mathbf{x}, \mathbf{d}, t_k) \rightarrow (c, \sigma). \tag{2}$$

We design the time offsets to be continuous rather than discrete frame indices. Further details are provided in the following subsections. The time offsets are jointly optimized with NeRF parameters by minimizing MSE between the ground truth RGB pixel $\mathbf{C}$ and the volume-rendered RGB pixel $\hat{\mathbf{C}}$:

$$\mathcal{L}_{\text{RGB}} = \sum_{k,\mathbf{r},t} \left\| \hat{\mathbf{C}}(\mathbf{r}, t + \delta_k) - \mathbf{C}_k(\mathbf{r}, t) \right\|_2^2, \tag{3}$$

where $k, \mathbf{r}, t$ are the camera index, center ray, and time of each pixel in the training frames, respectively. To calculate $\hat{\mathbf{C}}$, we use the same numerical quadrature that approximates volume rendering integral as previous papers (Max 1995; Mildenhall et al. 2021). The time offsets resolve the disagreement across multi-view supervisions and significantly improve the reconstruction quality, especially on the dynamic parts.

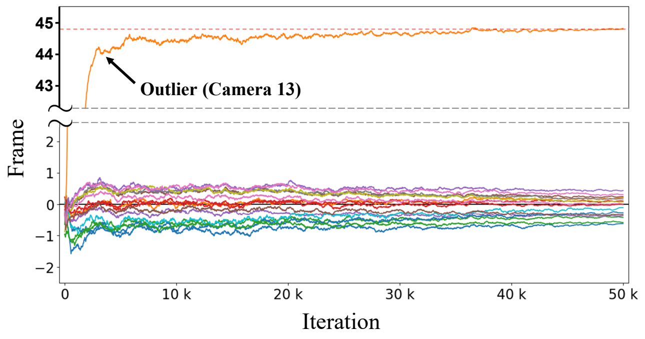


Figure 3: Learning curve of time offsets. We show camera offsets in `coffee_martini` scene along the training iterations of Sync-MixVoxels. Our method successfully finds the offset of the outlier camera.

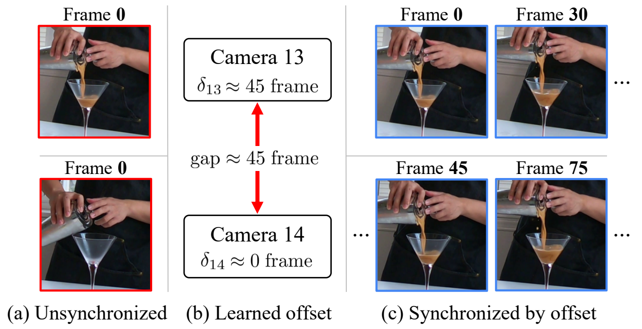
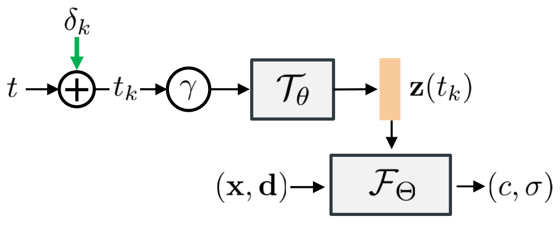


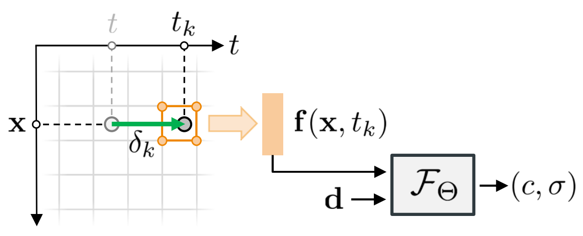
Figure 4: Synchronization with time offsets. (a) For given unsynchronized videos, (b) our method finds the time offsets $\delta$ which are equivalent to (c) automatically synchronizing the videos.

Figure 3 is an exemplar plot of the learned time offsets over training iterations on `coffee_martini`. This scene has an unsynchronized view as shown in Figure 1a. The time offsets are initialized as zero and converge to the visually correct offset. As the common scenes do not provide ground truth offsets, we report errors on synthetic dynamic scenes with the ground truth offsets in the following experiments. We note that the optimization of time offsets does not require additional loss functions or regularizers.

When we assess the reconstruction of the scene captured from a test view, even the test view can be unsynchronized. For rendering a video from a novel view, we can customize the time offset $\delta_{\text{test}}$. Our method allows to optimize $\delta_{\text{test}}$ with the frozen trained model such that the potential time offset of the test view can be resolved. To fully exploit the advantage of our method, we optimize the time offset for the test view when we report the test view performance.



(a) Implicit function-based approach



(b) Grid-based approach

Figure 5: Continuous temporal embedding. (a) We present an implicit function-based approach for the methods utilizing per-frame temporal embeddings. We add time offset $\delta_k$ of camera $k$ to time input $t$. $\mathcal{T}_\theta$ is the implicit function for mapping calibrated time into temporal embedding $\mathbf{z}$. (b) We query the embedding at the calibrated time $t_k$ on grid-based models. Bilinear interpolation naturally allows continuous temporal embedding.

## Implicit Function for Temporal Embedding

Various dynamic NeRFs (Park et al. 2021b; Li et al. 2022b; Wang et al. 2022a) explicitly learn latent embeddings of individual frames to represent the temporal variations of dynamic scenes. These latent embeddings do not represent the moments between frames nor are their interpolation is not guaranteed to produce smooth dynamics. Furthermore, the number of embeddings also increases as the video becomes longer, requiring more memory usage.

Instead of optimizing hundreds of individual embedding vectors for all frames, we train an implicit neural representation $\mathcal{T}_\theta$ that produces the temporal embedding $\mathbf{z}(t)$ for an arbitrary time input $t$ (Figure 5a). First, we encode a normalized time input $t$ using a set of sinusoidal functions:

$$\gamma(t, L) = \left[\sin(2^0\pi t), \cdots, \sin(2^{L-1}\pi t), \cos(2^{L-1}\pi t)\right]^{\mathrm{T}}. \tag{4}$$

Similar to previous observations (Mildenhall et al. 2021; Tancik et al. 2020), where inputs encoded with high-frequency functions lead to a better fit for high-frequency variations in data, this mapping assists $\mathcal{T}_\theta$ in capturing the movement of the scene. Thus, Eq. 2 is modified as follows:

$$\mathcal{F}_\Theta : (\mathbf{x}, \mathbf{d}, \mathbf{z}(t_k)) \rightarrow (\mathbf{c}, \sigma), \text{ where } \mathbf{z}(t) = \mathcal{T}_\theta(\gamma(t, L)), \tag{5}$$

where $t_k$ is the time input shifted by the per-camera time offset used in Eq. 2. To capture the rapid scene motion, we set $L = 10$ in all experiments. We select MixVoxels as a baseline for using per-frame temporal latents and compare it with our Sync-MixVoxels in Experiments Section.

## Grid-Based Models with Time Offsets

Grid-based dynamic NeRFs (Fridovich-Keil et al. 2023; Park et al. 2023; Attal et al. 2023) calculate latent vectors from the feature grid to feed the latents to the time-dependent function $\mathcal{F}_\Theta$. Specifically, for a given 4D coordinates $(\mathbf{x}, t)$, the latent representation $\mathbf{f}(\mathbf{x}, t)$ is obtained by linearly interpolating feature vectors assigned to each vertex of the grid to which the coordinates belong.

We use camera-specific time $t_k$ instead of the original time $t$. In grid-based models, Eq. 2 is modified as follows:

$$\mathcal{F}_\Theta : (\mathbf{f}(\mathbf{x}, t_k), \mathbf{d}) \rightarrow (\mathbf{c}, \sigma), \text{ where } \mathbf{f}(\mathbf{x}, t) = \text{Grid}(\mathbf{x}, t), \tag{6}$$

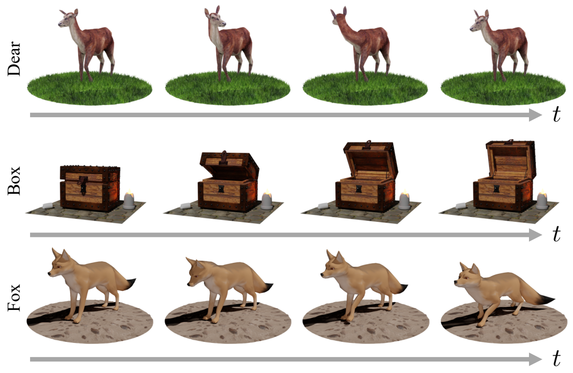


Figure 6: Snapshots of the Unsynchronized Dynamic Blender Dataset. We show exemplar frames of a video from our multi-view synthetic dataset.

and Grid$(\cdot)$ denotes the interpolation of grid vectors surrounding given coordinates. Figure 5b illustrates how our method modifies the sampling in the grid by $\delta_k$. We select K-Planes as a baseline for using grid-based representation and compare it with our Sync-K-Planes.

## Experiments

In this section, we validate the effectiveness of our method using MixVoxels and K-Planes as baselines. We begin by describing the datasets and evaluation metrics. We then show that our method significantly improves the baselines regarding the shape of moving objects and overall reconstruction. Next, we validate the accuracy of the found time offsets. Finally, we demonstrate the robustness of our method against different levels of unsynchronization including synchronized settings.

### Evaluation Settings

**Unsynchronized Datasets.** The Plenoptic Video Dataset (Li et al. 2022b) contains six challenging real-world scenes with varying degrees of dynamics. Its multi-view videos are roughly synchronized except `coffee_martini` scene. In order to simulate an in-the-wild unsynchronized capturing environment, we modify this dataset to be unsynchronized

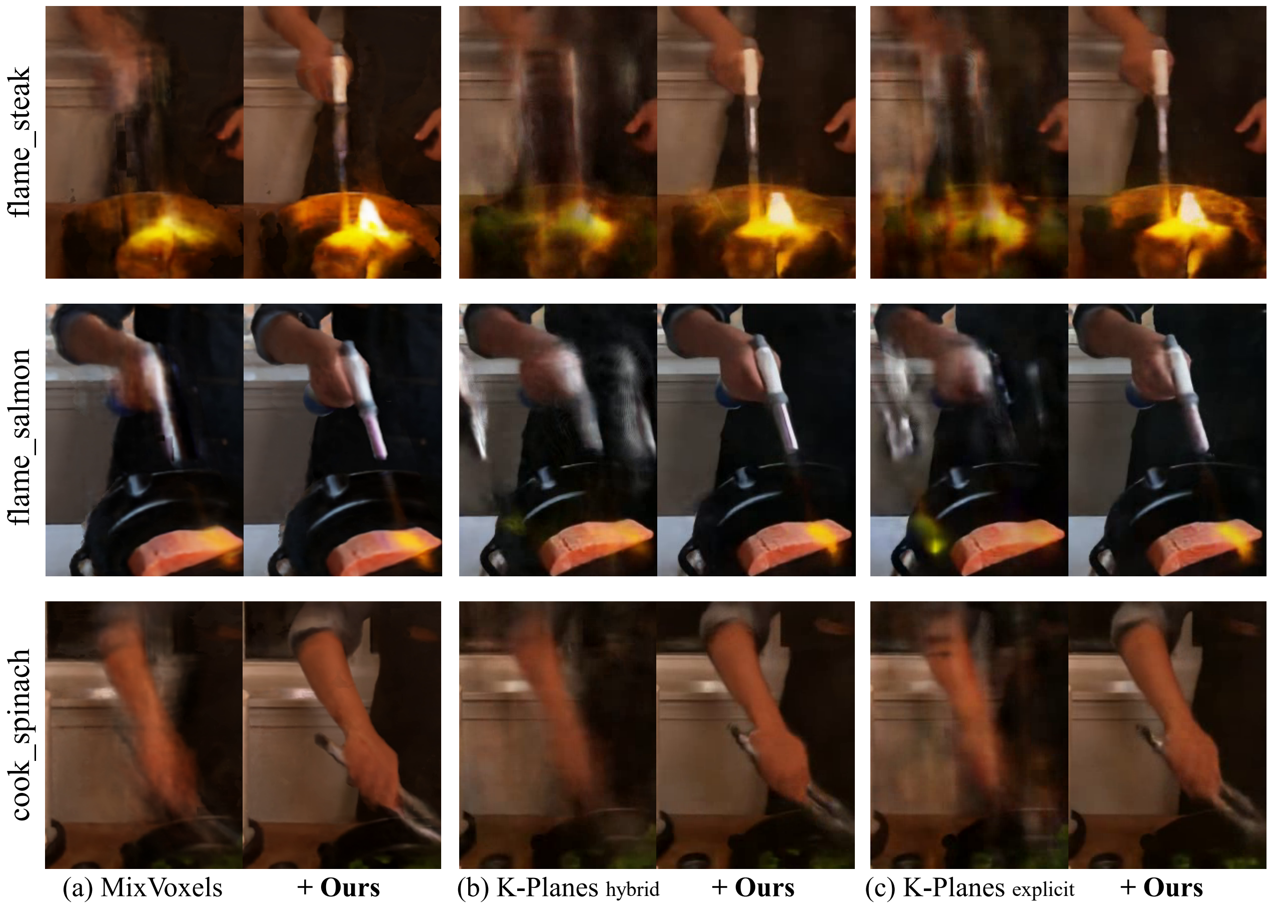


Figure 7: Cropped renderings on Unsynchronized Plenoptic Video Dataset. While the baselines produce severe artifacts, employing our method on them resolves the problem.

by randomly translating along the temporal axis. The time offsets are sampled from a normal distribution with zero mean and a standard deviation of 5, and then rounded to be integers. More details are in Appendix A.

Although the Plenoptic Video Dataset is synchronized, we cannot prepare ground truth offsets because the synchronization is not perfect. As a solution, we create an Unsynchronized Dynamic Blender Dataset as the following process. We start from free public Blender assets with motion, namely `box`, `fox`, and `deer`. These assets are rendered on similar camera setups. Subsequently, we translate the rendered videos according to random time offsets drawn from the aforementioned normal distribution. Then we have access to the ground truth time offsets because these rendered videos are perfectly synchronized. All videos are 10 seconds long and captured in 14 fixed frontal-facing multi-view cameras at a frame rate of 30 FPS following the Plenoptic Video Dataset. Example frames are shown in Figure 6. The dataset is publicly available.

**Baselines.** We employ MixVoxels and K-Planes as baselines and adopt our method upon them, namely, Sync-MixVoxels and Sync-K-Planes. We select the baselines because they are the latest among the methods with per-frame temporal latent and grid-based temporal representation.

**Evaluation Metrics.** We evaluate the rendering quality using the following quantitative metrics. To quantify the pixel color error, we report PSNR (peak signal-to-noise ratio) between rendered video and the ground truth. To consider perceived similarity, we report SSIM (Wang et al. 2004). To measure higher-level perceptual similarity, we report LPIPS (Zhang et al. 2018) using AlexNet and VGG. Higher values for PSNR and SSIM, and lower values for LPIPS indicate better visual quality. To measure the accuracy of the time offsets, the mean absolute error (MAE) between the found offsets and the ground truth offsets is measured in seconds.

## Rendering Quality

We evaluate the effectiveness of our method over the baselines by observing rendering quality. We highly recommend watching the result video on our project page.

**Unsynchronized Plenoptic Video Dataset.** Figure 7 compares ours with the baselines on the Unsynchronized Plenoptic Video Dataset. All the baselines produce severe artifacts in dynamic parts and perform even worse on

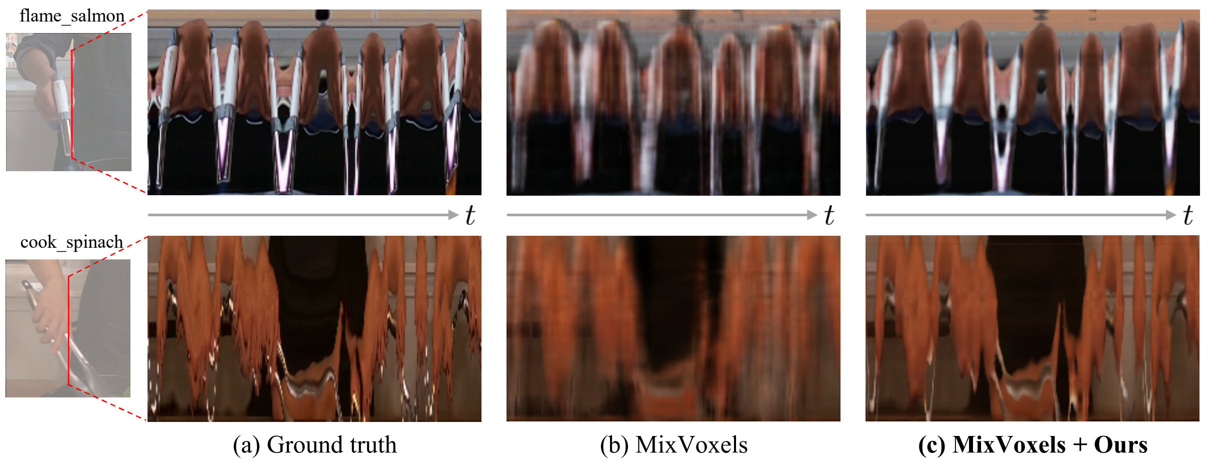


Figure 8: Spatiotemporal Images. Each column in the image represents the rendered pixels on the fixed vertical line at a moment. The fixed vertical line with 150 pixels is shown in the leftmost image patch. We render all frames in the video, producing 270-pixel-wide spatiotemporal images. Our results are much clearer than the baseline.

| model \ metric | PSNR | SSIM | LPIPS-A | LPIPS-V |
|---|---|---|---|---|
| MixVoxels | 29.96 | 0.9059 | 0.1669 | 0.2648 |
| Sync-MixVoxels | **30.53** | **0.9101** | **0.1570** | **0.2575** |
| K-Planes-H | 29.16 | 0.9120 | 0.1278 | 0.2222 |
| Sync-K-Planes-H | **30.44** | **0.9243** | **0.1064** | **0.1989** |
| K-Planes-E | 28.51 | 0.9042 | 0.1484 | 0.2438 |
| Sync-K-Planes-E | **29.97** | **0.9223** | **0.1144** | **0.2103** |

Table 1: Average performance in the *test view* on Unsynchronized Plenoptic Video Dataset. Our method improves all the baselines, even achieving performance similar to synchronized setting in Table 6. H stands for hybrid and E stands for explicit.

| model \ metric | PSNR | SSIM | LPIPS-A | LPIPS-V |
|---|---|---|---|---|
| MixVoxels | 31.13 | 0.9115 | 0.1565 | 0.2565 |
| Sync-MixVoxels | **31.87** | **0.9162** | **0.1457** | **0.2495** |
| K-Planes-H | 29.25 | 0.9013 | 0.1548 | 0.2477 |
| Sync-K-Planes-H | **30.59** | **0.9221** | **0.1118** | **0.2042** |
| K-Planes-E | 29.64 | 0.9095 | 0.1461 | 0.2386 |
| Sync-K-Planes-E | **30.79** | **0.9238** | **0.1147** | **0.2050** |

Table 2: Average performance over all *training views* on Unsynchronized Plenoptic Video Dataset. Our method improves all the baselines.

scenes with larger motion. Opposed to the reported performance of K-Planes outperforming MixVoxels on synchronized dataset, K-Planes suffers more on dynamic parts in unsynchronized setting: the hand and the torch are rendered in multiple places in K-Planes. Refer to per-scene results in Appendix F.1.

However, our method successfully corrects artifacts in both baselines. Additionally, in Figure 8, we visually demonstrate the comparison of spatio-temporal images on dynamic regions. Each column in an image represents the rendered rays on the fixed vertical line at a time. Horizontally concatenating columns of all frames from the test view constructs a spatio-temporal image as a whole. Our method exhibits significantly clearer spatio-temporal images compared to the baseline (Figure 8b-c).

Table 1 reports quantitative metrics on the *test* views of the unsynchronized Plenoptic Video Dataset. We report the average across all scenes and refer per-scene performance to the Appendix F.1. The above values are the result after optimizing the time offset in the test view. The previous results are reported in Appendix B.

Table 2 reports the same metrics on the *training* views. The baselines struggle to reconstruct even the training views. On the other hand, adopting our method on the baselines fixes the problem.

**Unsynchronized Dynamic Blender Dataset.** Figure 9 compares the baselines and our approaches on `fox` scene from Unsynchronized Dynamic Blender Dataset. While MixVoxels and K-Planes struggle on the motion and object boundaries, our Sync-MixVoxels and Sync-K-Planes shows the clearer face of the fox.

Table 3 reports quantitative metrics on the *test* views of the Unsynchronized Dynamic Blender Dataset. We report the average across all scenes and refer to per-scene performance in Appendix F.2. Our method improves the baselines in all cases.

## Time Offset Accuracy

We demonstrate that the time offsets found by our method are highly accurate. Table 4 reports the mean absolute error (MAE) between the optimized time offsets and the ground truth offsets on the Unsynchronized Dynamic Blender Dataset. The average error is approximately 0.01 seconds, which corresponds to less than one-third of a frame.

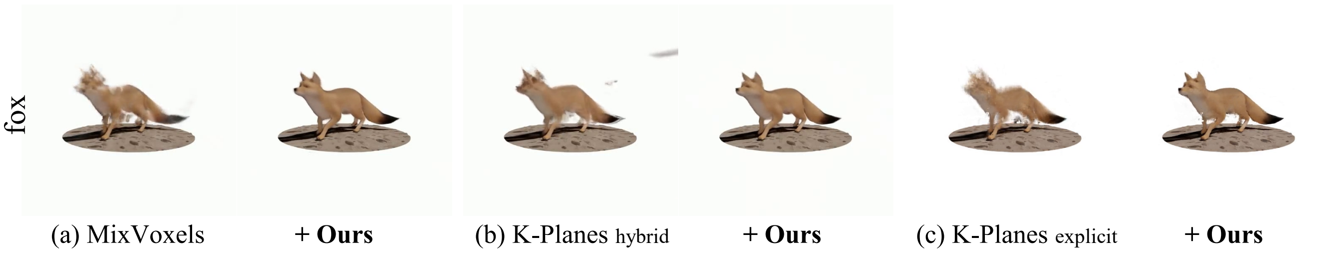


Figure 9: Qualitative results on Unsynchronized Dynamic Blender Dataset. Visual comparison of MixVoxels, K-Planes hybrid, K-Planes explicit and our method on them with `fox` scene.

| model \ metric | PSNR | SSIM | LPIPS-A | LPIPS-V |
|---|---|---|---|---|
| MixVoxels | 31.06 | 0.9753 | 0.0237 | 0.0339 |
| Sync-MixVoxels | **37.11** | **0.9841** | **0.0120** | **0.0226** |
| K-Planes-H | 32.66 | 0.9771 | 0.0175 | 0.0244 |
| Sync-K-Planes-H | **39.40** | **0.9863** | **0.0066** | **0.0131** |
| K-Planes-E | 32.15 | 0.9751 | 0.0246 | 0.0335 |
| Sync-K-Planes-E | **39.35** | **0.9865** | **0.0072** | **0.0140** |

Table 3: Average performance in the *test view* on Unsynchronized Dynamic Blender Dataset. Our method improves all the baselines.

| | MAE (seconds) |
|---|---|
| Sync-MixVoxels | 0.0154 |
| Sync-K-Planes-H | 0.0156 |
| Sync-K-Planes-E | 0.0229 |

Table 4: MAE between predicted offset and ground truth on Unsynchronized Dynamic Blender Dataset. Our method accurately finds time offsets achieving MAE of approximately 0.01 seconds.

| | 1.5× | | 2.0× | |
|---|---|---|---|---|
| model \ scene | cook spinach | fox | cook spinach | fox |
| MixVoxels | 30.21 | 31.10 | 30.48 | 29.81 |
| Sync-MixVoxels | **31.44** | **36.15** | **31.50** | **35.58** |
| K-Planes-H | 29.93 | 31.85 | 29.13 | 29.71 |
| Sync-K-Planes-H | **31.71** | **40.31** | **31.85** | **40.36** |
| K-Planes-E | 28.93 | 31.46 | 28.46 | 29.10 |
| Sync-K-Planes-E | **31.12** | **40.47** | **31.16** | **40.29** |

Table 5: Average PSNR in the *test view* with varaious unsynchonization lengths. Our method is robust to different lengths of unsynchronization.

| model \ metric | PSNR | SSIM | LPIPS-A | LPIPS-V |
|---|---|---|---|---|
| MixVoxels | 30.39 | 0.9100 | 0.1577 | 0.2586 |
| Sync-MixVoxels | **30.41** | **0.9104** | **0.1559** | **0.2564** |
| K-Planes-H | 30.40 | **0.9257** | **0.1044** | **0.1980** |
| Sync-K-Planes-H | **30.56** | 0.9246 | 0.1059 | 0.1998 |
| K-Planes-E | 30.04 | 0.9229 | 0.1131 | 0.2083 |
| Sync-K-Planes-E | **30.14** | **0.9237** | **0.1121** | **0.2072** |

Table 6: Average performance in the *test view* on Synchronized Plenoptic Video Dataset. Our method enhances performance by resolving the small temporal gaps.

## Versatility to Various Scenarios

**Various Unsynchronization Lengths.** We evaluate the performance of all methods under different lengths of unsynchronization, namely $1.5\times$ and $2\times$ long offsets. Table 5 shows that our method consistently improves the performance of the baselines. We note that the unsynchronization deteriorates K-Planes more than MixVoxels even though K-Planes outperform MixVoxels on synchronized datasets. Nevertheless, our method successfully reflects their ranking in synchronized setting to unsynchronized setting.

**Synchronization Setting.** We verify that our method improves the baselines even on synchronized settings (Table 6). Although the gaps between the baselines and ours are smaller than unsynchronized setting, this improvement implies that the dataset assumed to be synchronized is not perfectly synchronized and even tiny offsets less than a frame are correctly found by our method.

## Conclusion

Our work is the first attempt to train dynamic NeRFs on unsynchronized multi-view videos. We have shown that the existing dynamic NeRFs deteriorate when the videos are not synchronized. As its reason lies in the previous method using a single temporal latent embedding for a multi-view frame, we introduce time offsets for individual views such that the videos can be synchronized by the offsets. We jointly optimize the offsets along with NeRF, using a typical reconstruction loss. Our method, Sync-NeRF, is versatile to various types of existing dynamic NeRFs, including MixVoxels, K-Planes, and more. It consistently improves the reconstruction on both unsynchronized and synchronized settings.

**Discussion** Although our method drives dynamic NeRFs closer to in-the-wild captures, we still lack the means to generalize to the causal handheld cameras. We suggest it as an interesting future research topic. While our method does not introduce additional computational complexity to grid-based temporal embedding, implicit function-based temporal embeddings require an additional training time and memory for training the function. Nevertheless, in the inference phase, the temporal embeddings can be pre-computed for all frames leading to negligible overhead.

## Acknowledgments

This work is supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00072, Development of Audio/Video Coding and Light Field Media Fundamental Technologies for Ultra Realistic Tera-media)

## References

Attal, B.; Huang, J.-B.; Richardt, C.; Zollhoefer, M.; Kopf, J.; O'Toole, M.; and Kim, C. 2023. HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16610–16620.

Cao, A.; and Johnson, J. 2023. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 130–141.

Fang, J.; Yi, T.; Wang, X.; Xie, L.; Zhang, X.; Liu, W.; Nießner, M.; and Tian, Q. 2022. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. In *SIGGRAPH Asia 2022 Conference Papers*.

Fridovich-Keil, S.; Meanti, G.; Warburg, F. R.; Recht, B.; and Kanazawa, A. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12479–12488.

Gao, H.; Li, R.; Tulsiani, S.; Russell, B.; and Kanazawa, A. 2022. Monocular Dynamic View Synthesis: A Reality Check. In *NeurIPS*.

Jeong, Y.; Ahn, S.; Choy, C.; Anandkumar, A.; Cho, M.; and Park, J. 2021. Self-Calibrating Neural Radiance Fields. In *Proceedings of the Int. Conf. on Computer Vision (ICCV)*.

Li, L.; Shen, Z.; Wang, Z.; Shen, L.; and Tan, P. 2022a. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems*, 35.

Li, T.; Slavcheva, M.; Zollhoefer, M.; Green, S.; Lassner, C.; Kim, C.; Schmidt, T.; Lovegrove, S.; Goesele, M.; Newcombe, R.; et al. 2022b. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Li, Z.; Wang, Q.; Cole, F.; Tucker, R.; and Snavely, N. 2023. DynIBaR: Neural Dynamic Image-Based Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, C.-H.; Ma, W.-C.; Torralba, A.; and Lucey, S. 2021. BARF: Bundle-Adjusting Neural Radiance Fields. In *IEEE International Conference on Computer Vision (ICCV)*.

Liu, Y.-L.; Gao, C.; Meuleman, A.; Tseng, H.-Y.; Saraf, A.; Kim, C.; Chuang, Y.-Y.; Kopf, J.; and Huang, J.-B. 2023. Robust Dynamic Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Martin-Brualla, R.; Radwan, N.; Sajjadi, M. S. M.; Barron, J. T.; Dosovitskiy, A.; and Duckworth, D. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*.

Max, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2): 99–108.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

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

Park, K.; Sinha, U.; Hedman, P.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Martin-Brualla, R.; and Seitz, S. M. 2021b. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*.

Park, S.; Son, M.; Jang, S.; Ahn, Y. C.; Kim, J.-Y.; and Kang, N. 2023. Temporal Interpolation Is All You Need for Dynamic Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4212–4221.

Pumarola, A.; Corona, E.; Pons-Moll, G.; and Moreno-Noguer, F. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *arXiv preprint arXiv:2011.13961*.

Shrstha, P.; Barbieri, M.; and Weda, H. 2007. Synchronization of multi-camera video recordings based on audio. In *Proceedings of the 15th ACM international conference on Multimedia*, 545–548.

Song, L.; Chen, A.; Li, Z.; Chen, Z.; Chen, L.; Yuan, J.; Xu, Y.; and Geiger, A. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5): 2732–2742.

Tancik, M.; Casser, V.; Yan, X.; Pradhan, S.; Mildenhall, B.; Srinivasan, P. P.; Barron, J. T.; and Kretzschmar, H. 2022. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8248–8258.

Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; and Ng, R. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547.

Wang, F.; Tan, S.; Li, X.; Tian, Z.; and Liu, H. 2022a. Mixed neural voxels for fast multi-view video synthesis. *arXiv preprint arXiv:2212.00190*.

Wang, L.; Zhang, J.; Liu, X.; Zhao, F.; Zhang, Y.; Zhang, Y.; Wu, M.; Yu, J.; and Xu, L. 2022b. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13524–13534.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhao, F.; Yang, W.; Zhang, J.; Lin, P.; Zhang, Y.; Yu, J.; and Xu, L. 2022. HumanNeRF: Efficiently Generated Human Radiance Field From Sparse Inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7743–7753.