Sync-NeRF: Generalizing Dynamic NeRFs to Unsynchronized Videos

Anonymous submission

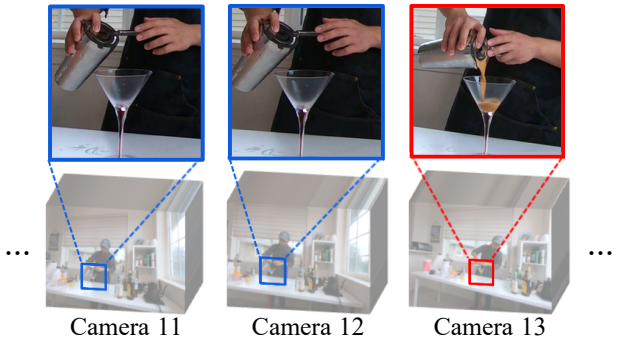
Abstract

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 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 and robustness of our method. The source code and the dataset will be released online.

1 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 which receives 3D coordinates and produces color and density, it adding time in its input inherently generalizes to multi-view videos to design dynamic NeRFs. 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 approaches for synchronization, such as timecode systems or audio peaks, usually work, these processes require additional device, cannot be applied on noisy environments, or can be inaccurate[1]. 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 if it is included in the training view as shown in 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

(b) MixVoxels **(c) MixVoxels + Ours**

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 synchronized, there can be temporal mismatch within a frame. Figure 2b illustrates the

[1](Li et al. 2022b) Appendix B: "We removed a particular camera if the time synchronization did not work."

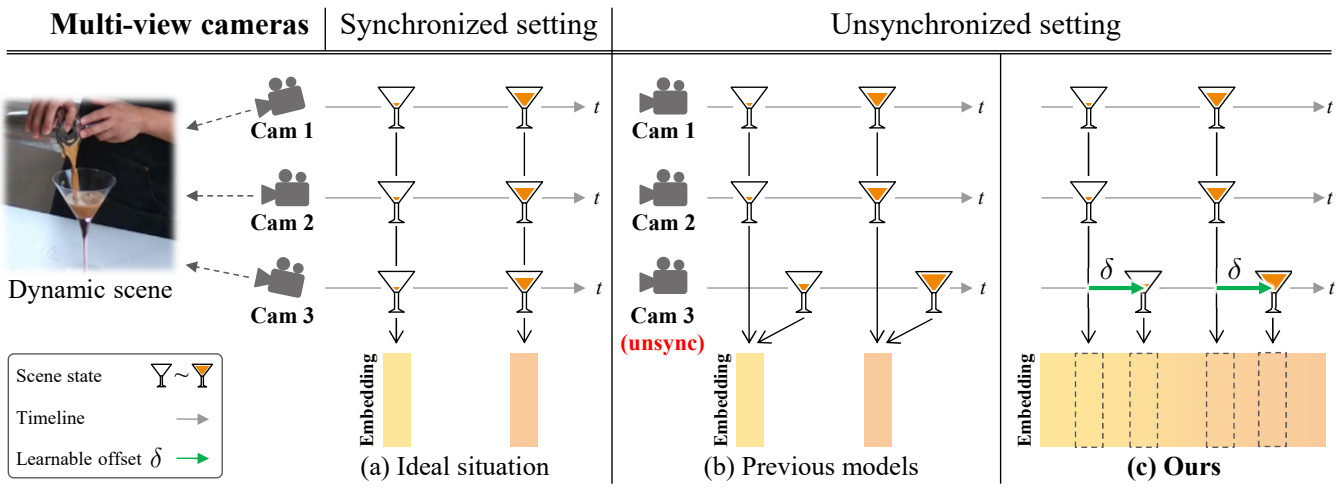


Figure 2: **Problem statement.** (a) Ideally, all multi-view images at a frame captures the same moment of a scene. Each frame is represented by a latent embedding. (b) Some frames are not synchronized. Previous methods suffer from the discrepancy between the latent embedding of the frame and the actual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

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. To apply our method on dynamic NeRFs with discrete temporal representation (Wang et al. 2022a), we further design a continuous function that receives time and produces temporal representation. For dynamic NeRFs with spatiotemporal planes (Fridovich-Keil et al. 2023), we inherit bilinear interpolation.

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.

2 Related Work

Dynamic NeRFs have been evolving by introducing better representation for specific purpose. D-NeRF or HumanNeRF 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 they 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 easily adapted to existing methods.

On the other hand, the methods with 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 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 to reconstruct multi-view images 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 model parameters jointly to eliminate the requirements of known camera parameters. RoDyNeRF (Liu et al. 2023) optimizes a dynamic NeRF jointly with camera parameters to tackle the failure of COLMAP in highly dynamic scenes. However, previ-

ous methods can not compensate for the inaccurate synchronization across multi-view videos capturing dynamic scenes. Traditional approaches for synchronization utilize timecode systems (Li et al. 2022b) or audio peaks recorded simultaneously with videos (Shrstha, Barbieri, and Weda 2007). Therefore, these methods need 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.

3 Method

In this section, we introduce per-camera time offsets, which enable training dynamic NeRFs on the unsynchronized multi-view videos (Section 3.1). Subsequently, we describe an implicit function-based approach for models with per-frame temporal embeddings (Section 3.2), and an interpolation-based approach for grid-based models (Section 3.3).

3.1 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 Θ 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 δ_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 deferred to Section 3.2 and 3.3. 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). It resolves the disagreement across multi-view supervisions and significantly improves reconstruction, especially on the dynamic parts.

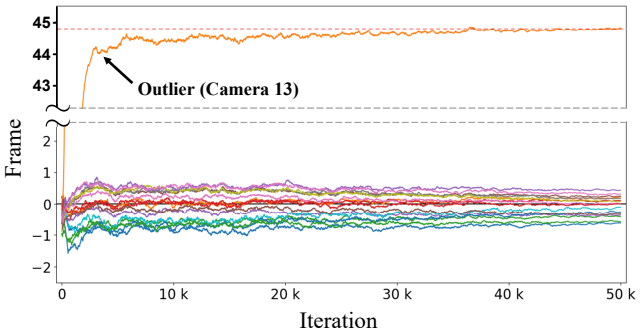


Figure 3: **Learning curve of time offsets.** We show camera offsets in `coffee_martini` scene along the training iterations of Sync-MixVoxels. Our method successfully finds the offset of the outlier camera.

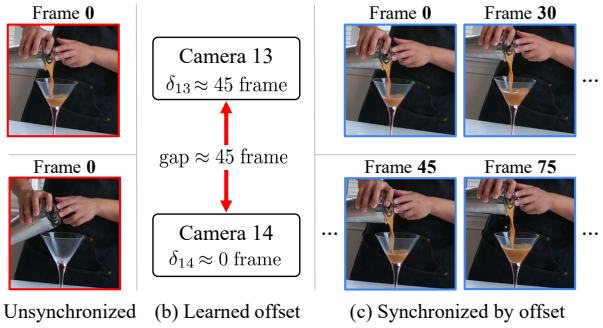
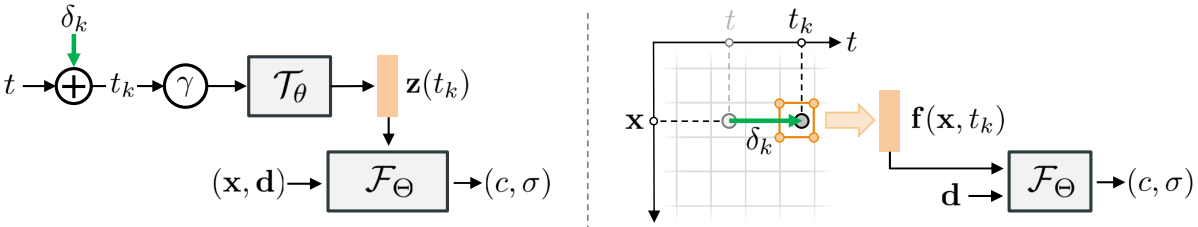


Figure 4: **Synchronization with time offsets.** (a) For given unsynchronized videos, (b) our method finds the time offsets δ which are equivalent to (c) automatically synchronizing the videos.

Figure 3 is an exemplar plot of the learned time offsets over training iteration on `coffee_martini`. This scene has an unsynchronized view as shown in Figure 1a. The time offsets are initialized to 0 and converge to the visually correct offset. As the common scenes do not provide the ground truth offsets, we report errors on synthetic dynamic scenes with arbitrary ground truth offsets in the experiments. We note that it does not require additional loss functions or regularizers.

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



(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 δ_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.

3.2 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 increases for longer videos.

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 using high-frequency functions enable a better fit for high-frequency variations in the 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 per-frame temporal latents and compare it with our Sync-MixVoxels in Section 4.

3.3 Grid-based Models with Time Offsets

Grid-based dynamic NeRFs (Fridovich-Keil et al. 2023; Park et al. 2023; Attal et al. 2023) calculates latent vectors from the feature grid to feed them 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:

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

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

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

4 Experiments

In this section, we validate the effectiveness of our method using MixVoxels and K-Planes as baselines. Section 4.1 describes the datasets and evaluation metrics. Section 4.2 shows that our method significantly improves the baselines regarding the shape of moving objects and overall reconstruction. Section 4.3 validates the accuracy of the found time offsets. Section 4.4 demonstrates the robustness of our method against different levels of unsynchronization including synchronized settings.

4.1 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. 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 renderings of 3D 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 will be publicly available.

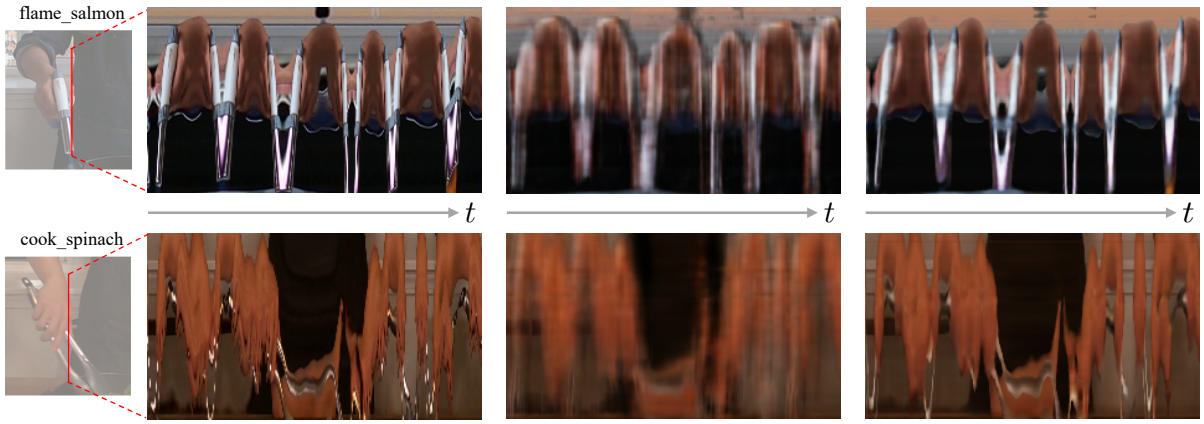
Baselines. We employ MixVoxels and K-Planes as our baselines and adopt our method upon them, namely, Sync-MixVoxels and Sync-K-Planes, respectively. They are the latest among the methods with per-frame temporal latent and grid-based temporal representation.

Evaluation Metrics. We evaluate the rendering quality in test views 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 VGG and AlexNet. Higher values for PSNR and SSIM, and lower values for LPIPS indicate better visual quality. Last but not least, the mean absolute error (MAE) of the found time offsets are measured in seconds.

4.2 Rendering quality

We evaluate the effectiveness of our method on the baselines by observing rendering quality. We highly recommend watching the supplementary video.

Unsynchronized Plenoptic Video Dataset. Figure 7 compares ours with the baselines on the Unsynchronized Plenoptic Video Dataset. All the baselines produce severe artifacts on dynamic parts such as hand, torch, and tongs. 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-



(a) Ground truth (b) MixVoxels **(c) MixVoxels + Ours**

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.

Planes. The baselines suffer worse on scenes with larger motion, namely `cook_spinach` > `cut_roasted_beef` > `flame_steak` > `flame_salmon` > `coffee_martini` > `sear_steak`. Refer to per-scene results in Appendix F.1.

Above all, our method successfully corrects artifacts in both baselines. Additionally, in Figure 8, we visually demonstrate the performance of our method 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 spatiotemporal image as a whole. Compared to the baseline, our method exhibits significantly clearer spatialtemporal images (Figure 8b-c).

model \ metric		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
MixVoxels		29.96	0.9059	0.1669	0.2648
Sync-MixVoxels		**30.53**	**0.9101**	**0.1570**	**0.2575**
K-Planes	hybrid	29.16	0.9120	0.1278	0.2222
Sync-K-Planes	hybrid	**30.44**	**0.9243**	**0.1064**	**0.1989**
K-Planes	explicit	28.51	0.9042	0.1484	0.2438
Sync-K-Planes	explicit	**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.

Table 1 reports quantitative metrics on the *test* views of the unsynchronized Plenoptic Video Dataset. We report the average across all scenes and defer per-scene performance to the Appendix F.1. Our method improves the baselines in all cases nearly to the performance on synchronized setting (Section 4.4). 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 baseline struggles to reconstruct even the training views. It shows the challenges due to more unsynchronized views. On the other hand, adopting our method on the baselines fixes the problem.

model \ metric		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
MixVoxels		31.13	0.9115	0.1565	0.2565
Sync-MixVoxels		**31.87**	**0.9162**	**0.1457**	**0.2495**
K-Planes	hybrid	29.25	0.9013	0.1548	0.2477
Sync-K-Planes	hybrid	**30.59**	**0.9221**	**0.1118**	**0.2042**
K-Planes	explicit	29.64	0.9095	0.1461	0.2386
Sync-K-Planes	explicit	**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.

Unsynchronized Dynamic Blender Dataset. Figure 9 compares the baselines and ours on `fox` scene from Unsynchronized Dynamic Blender Dataset. While MixVoxels struggles on the motion and object boundaries, Sync-MixVoxels show clear face of the fox. Surprisingly both variants of K-Planes fail drastically. Nevertheless, Sync-K-Planes resolves the problem.

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

4.3 Time Offset Accuracy

In this section, we demonstrate that the time offsets found by our method are highly accurate. Table 4 reports mean absolute error (MAE) of the optimized time offsets compared to ground truth time 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.

(a) MixVoxels + **Ours** (b) K-Planes hybrid + **Ours** (c) K-Planes explicit + **Ours**

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$_{alex}$	LPIPS$_{vgg}$
MixVoxels		30.62	0.9652	0.0364	0.0467
Sync-MixVoxels		**33.74**	**0.9749**	**0.0272**	**0.0386**
K-Planes	hybrid	21.76	0.8438	0.2665	0.2676
Sync-K-Planes	hybrid	**34.18**	**0.9700**	**0.1084**	**0.0482**
K-Planes	explicit	21.35	0.8617	0.2128	0.2251
Sync-K-Planes	explicit	**32.89**	**0.9629**	**0.0420**	**0.0680**

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

		MAE (seconds)
Sync-MixVoxels		0.0129
Sync-K-Planes	hybrid	0.0141
Sync-K-Planes	explicit	0.0165

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.

4.4 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.

		1.5×			2.0×		
model \ scene		cook spinach	flame salmon	fox	cook spinach	flame salmon	fox
MixVoxels		30.21	27.27	27.08	30.48	27.66	30.52
Sync-MixVoxels		**31.44**	**28.59**	**35.54**	**31.50**	**28.74**	**34.64**
K-Planes	hybrid	29.93	26.11	19.62	29.13	26.01	16.99
Sync-K-Planes	hybrid	**31.71**	**28.67**	**28.07**	**31.85**	**28.22**	**32.57**
K-Planes	explicit	28.93	24.71	20.52	28.46	24.83	17.33
Sync-K-Planes	explicit	**31.12**	**28.41**	**25.04**	**31.16**	**27.09**	**28.15**

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

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

model \ metric		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
MixVoxels		30.39	0.9100	0.1577	0.2586
Sync-MixVoxels		**30.41**	**0.9104**	**0.1559**	**0.2564**
K-Planes	hybrid	30.40	**0.9257**	**0.1044**	**0.1980**
Sync-K-Planes	hybrid	**30.56**	0.9246	0.1059	0.1998
K-Planes	explicit	30.04	0.9229	0.1131	0.2083
Sync-K-Planes	explicit	**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 on synchronized setting by resolving the small temporal gaps.

5 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 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 with NeRF with typical reconstruction loss. Our method, Sync-NeRF, is versatile to different types of existing dynamic NeRFs to build Sync-MixVoxels and Sync-K-Planes, and consistently improves their performance on both synchronized and unsynchronized settings.

Discussion Despite our method drives dynamic NeRFs closer to in-the-wild captures, we still lack the means to generalize to 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.

Supplemental Material for Sync-NeRF

Anonymous submission

A Additional Details

A.1 Training Details

Sync-MixVoxels The implicit function $\mathcal{T}_\theta$ is a 2-layer multi-layer perceptron (MLP) with 512 hidden units. We use a batch size of 128 rays querying all frames of input video because of memory resource constraints. We train both MixVoxels and Sync-MixVoxels for 60K iterations. The time offset δ_k for Sync-MixVoxels starts training after the first 10K for initial training stability. We set the learning rate of implicit function and time offset to be $5\times$ and $0.5\times$ of the MixVoxels backbone, respectively. The rest of the details follow the MixVoxels paper.

Sync-K-Planes We train K-Planes with a batch size of 4096 rays. Two versions of K-Planes, hybrid and explicit, are trained for 90K and 120K iterations, respectively. K-Planes applies ray sampling based on the temporal difference (IST) weight map proposed by DyNeRF. We apply the ISG sampling strategy for the first 50K iterations with time offset and then apply the IST strategy for both K-Planes and Sync-K-Planes. We set the time offset learning rate to be $0.1\times$ of the K-Planes. We roughly change the normalization range of time t to $[-0.8, 0.8]$ so that the camera-specific time $t_k = t + \delta_k$ does not deviate from the feature planes. For a comparison of quantitative results for the temporal axis resolution of the K-Planes, see Section E.

Misc. All experiments are trained on a single NVIDIA A5000. Since the target datasets are forward-facing scenes, we use normalized device coordinates.

A.2 Dataset Details

Dynamic Blender Dataset To create a dynamic Blender dataset, we use free 3D models that provide animation[1]. We add a floor under the object for each scene and slightly modify the animation to render it 300 frames at 512×512 resolution. We do not apply downsampling for training. 13 cameras are uniformly sampled on a spherical surface to render a forward-facing scene, and one camera is placed in the center of the surface to be used as a test view. All cameras share intrinsic parameters.

Misc. To create unsynchronized datasets, we translate each video along the temporal axis and then use the first 270 frames of overlapping intervals as training data. In Section 4.4, we use the first 255 and 240 frames for $1.5\times$ and $2.0\times$ settings, respectively.

[1] `dear`: https://www.turbosquid.com/3d-models/3d-deer-animation-model/1012108
`box`: https://www.turbosquid.com/3d-models/treasure-box-candles-model-2010834
`fox`: https://www.turbosquid.com/3d-models/animation-3d-model-1589869

B Test View Offset Optimization

To fully exploit the advantages of camera-specific time offset, we optimize the test view time offset δ_{test} to evaluate our models. We optimize the time offset for 200 and 1K iterations on MixVoxels and K-Planes, respectively, and then perform the evaluation. We train only the time offset while freezing the entire network. Table S1 shows that optimizing the time offset in the test view improves performance slightly.

average	PSNR	SSIM	$\text{LPIPS}_{\text{alex}}$	$\text{LPIPS}_{\text{vgg}}$
Sync-MixVoxels	**30.53**	**0.9101**	**0.1570**	**0.2575**
⌞w/o δ_{test}	30.29	0.9091	0.1574	0.2579
Sync-K-Planes hybrid	**30.44**	**0.9243**	**0.1064**	**0.1989**
⌞w/o δ_{test}	30.25	0.9235	0.1067	0.1992
Sync-K-Planes explicit	**29.97**	**0.9223**	**0.1144**	**0.2103**
⌞w/o δ_{test}	29.81	0.9214	0.1146	0.2104

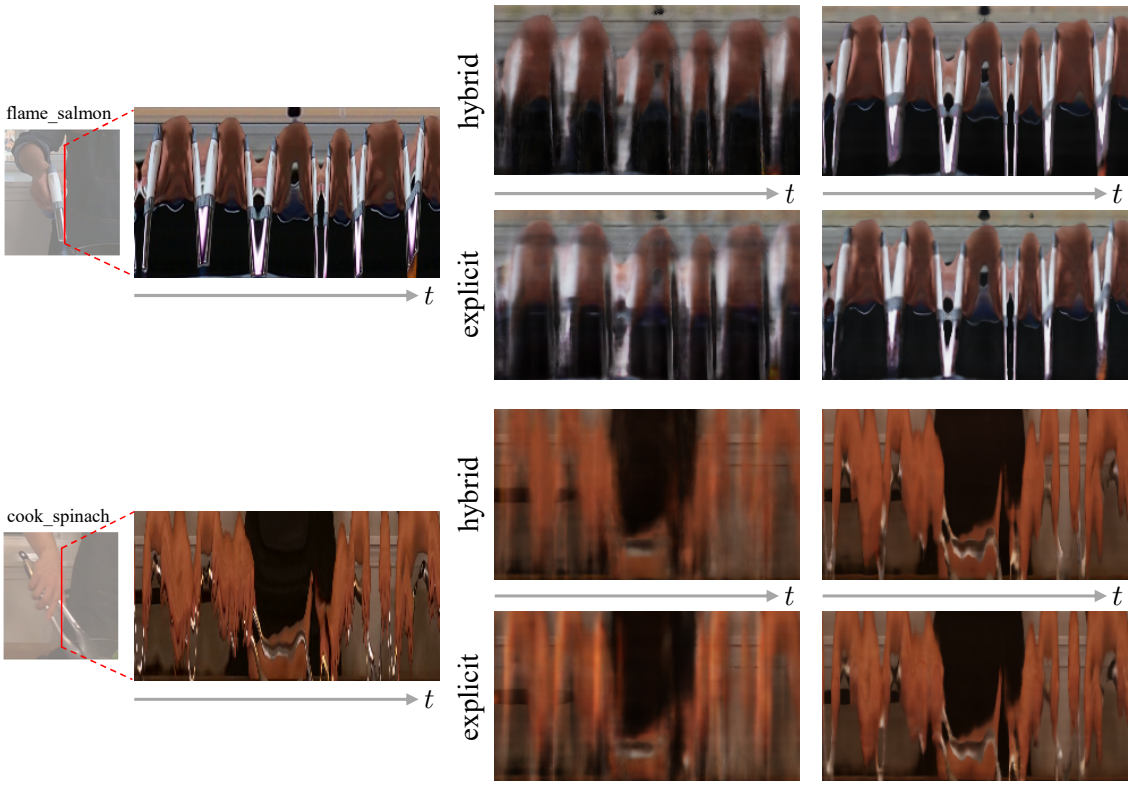
Table S1: **Comparison with test view offset optimization on Unsynchronized Plenoptic Video Dataset.** Test view offset optimization enables more accurate evaluation and improves our quantitative results.

C Spatiotemporal Image of K-Planes

In Figure S1, we show the spatiotemporal images for K-Planes and Sync-K-Planes on `flame_salmon` scene and `cook_spinach` scene. Applying our method, both hybrid and explicit versions of K-Planes render much clearer results.

D Sinusoidal Functions for Time

For our implicit function-based approach, we encode normalized time input using a set of sinusoidal functions to better represent the movement of the scene. Table S2 show the



(a) Ground truth (b) K-Planes **(c) K-Planes + Ours**

Figure S1: **Spatiotemporal Images of K-Planes.** Our results are much clearer than the baseline.

results of Sync-MixVoxels on `flame_salmon` scene for different Ls. With L=0, the model fails to represent dynamic scenes, and L=5 has a lower quality of dynamic regions than L=10, so we set L=10 as the default.

L for time	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
0	22.26	0.8411	0.2336	0.3001
5	28.80	0.8784	**0.2006**	0.2741
10 (default)	**28.85**	**0.8793**	0.2022	**0.2740**

Table S2: **Comparision of different Ls for sinusoidal functions on `flame_salmon` scene.** Using a high L value provides better dynamic region quality, so we set $L = 10$ as the default value.

E Temporal Axis Resolution of K-Planes

In our K-Planes experiments, the resolution of the temporal axis is lowered because the normalization range of time is reduced from $[-1.0, 1.0]$ to $[-0.8, 0.8]$. In Table S3, we report the results of increasing the temporal-axis resolution of the feature grid up to its original setting. As shown in the table, reducing the normalization range does not harm the overall performance.

	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
K-Planes $_{hybrid}$	26.19	0.8757	0.1698	**0.2559**
⌞ t-axis res ↑	**26.56**	**0.8760**	**0.1694**	0.2589
K-Planes $_{explicit}$	**25.61**	**0.8675**	**0.1891**	**0.2728**
⌞ t-axis res ↑	25.31	0.8631	0.1963	0.2821
Sync-K-Planes $_{hybrid}$	**29.17**	**0.9046**	**0.1259**	**0.2112**
⌞ t-axis res ↑	28.95	0.9027	0.1290	0.2131
Sync-K-Planes $_{explicit}$	28.56	**0.9048**	**0.1295**	**0.2114**
⌞ t-axis res ↑	**28.71**	0.9029	0.1302	0.2116

Table S3: **Comparison of increasing the temporal axis resolution on `flame_salmon` scene.** We observe that reducing the normalization range of time does not harm the overall performance.

F More Results

F.1 Plenoptic Video Dataset

We have demonstrate that our method significantly improves the baseline in the *test view* across all scenes of Unsynchronized Plenoptic Video Dataset. Figure S3 compares the baselines and ours on `coffee_martini`, `cut_rosted_beef` and `sear_steak` scene.

We report per-scene test view results in Table S4, and the average *train view* results in Table S5. Per-scene *test view* results on Synchronized Plenoptic Video Dataset are in Table S6.

(a) MixVoxels + **Ours** (b) K-Planes hybrid + **Ours** (c) K-Planes explicit + **Ours**

Figure S2: **More results on Unsynchronized Dynamic Blender Dataset.** Visual comparison of MixVoxels, K-Planes hybrid, K-Planes explicit and our method on them with `box` and `dear` scene.

F.2 Unsynchronized Dynamic Blender Dataset

Figure S2 compares ours with the baselines on Unsynchronized Dynamic Blender Dataset. On the `Box` scene, MixVoxels does not properly decompose the dynamic region, leading to some artifacts near the dynamic part on our Sync-MixVoxels. K-Planes surprisingly fails to fit the scene properly, causing deteriorated performance. On the other hand, our method shows a clear reconstruction of the box.

Table S7 reports the per-scene performance on the test views of the Unsynchronized Dynamic Blender Dataset. Our method significantly improves all the baseline, and it is comparable to the results in synchronized setting (Table S8).

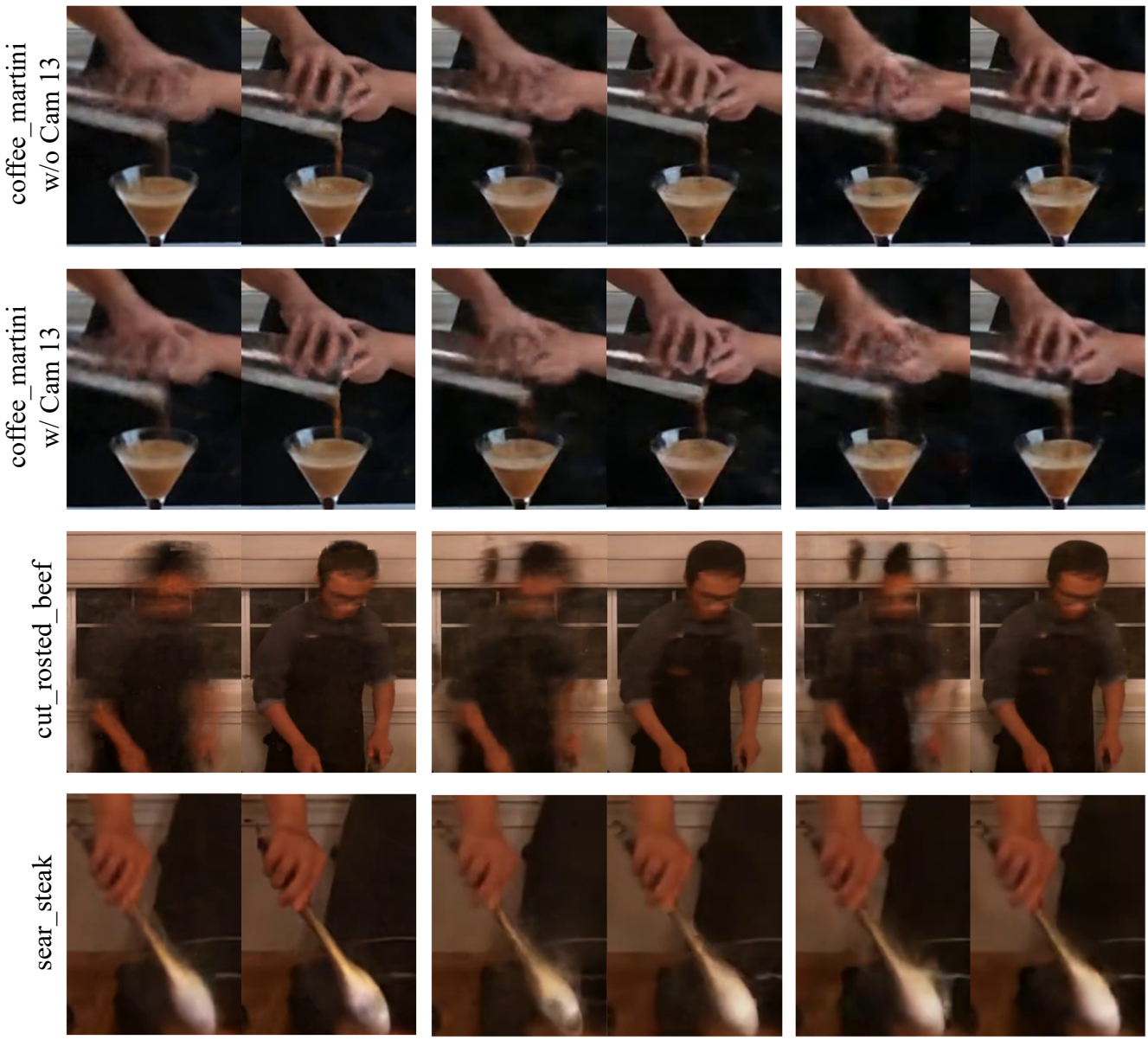


Figure S3: **More renderings on Unsynchronized Plenoptic Video Dataset.** While the baselines produce severe artifacts, employing our method on them resolves the problem.

test view	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
coffee_martini w/o Camera 13					**coffee_martini w/ Camera 13**				
MixVoxels	28.75	0.8831	0.1906	0.2752	MixVoxels	28.69	0.8817	0.1932	0.2767
Sync-MixVoxels	**28.96**	**0.8863**	**0.1843**	**0.2697**	Sync-MixVoxels	**29.06**	**0.8864**	**0.1861**	**0.2716**
K-Planes hybrid	28.29	0.8932	0.1446	0.2356	K-Planes hybrid	28.16	0.8944	0.1438	0.2347
Sync-K-Planes hybrid	**28.34**	**0.9026**	**0.1281**	**0.2158**	Sync-K-Planes hybrid	**28.61**	**0.9011**	**0.1343**	**0.2238**
K-Planes explicit	27.33	0.8807	0.1722	0.2614	K-Planes explicit	27.14	0.8793	0.1763	0.2656
Sync-K-Planes explicit	**28.30**	**0.9014**	**0.1369**	**0.2248**	Sync-K-Planes explicit	**27.88**	**0.8957**	**0.1475**	**0.2361**
cook_spinach					**cut_rosted_beef**				
MixVoxels	31.28	0.9191	0.1544	0.2659	MixVoxels	31.24	0.9209	0.1519	0.2633
Sync-MixVoxels	**31.94**	**0.9246**	**0.1381**	**0.2541**	Sync-MixVoxels	**31.67**	**0.9253**	**0.1358**	**0.2520**
K-Planes hybrid	30.05	0.9224	0.1170	0.2179	K-Planes hybrid	31.07	0.9266	0.1120	0.2124
Sync-K-Planes hybrid	**32.19**	**0.9382**	**0.0890**	**0.1890**	Sync-K-Planes hybrid	**32.15**	**0.9361**	**0.0911**	**0.1909**
K-Planes explicit	29.44	0.9187	0.1302	0.2297	K-Planes explicit	30.01	0.9185	0.1328	0.2405
Sync-K-Planes explicit	**31.27**	**0.9353**	**0.0981**	**0.1989**	Sync-K-Planes explicit	**31.56**	**0.9326**	**0.1026**	**0.2088**
flame_salmon					**flame_steak**				
MixVoxels	27.75	0.8750	0.2060	0.2769	MixVoxels	31.14	0.9286	0.1391	0.2508
Sync-MixVoxels	**28.85**	**0.8793**	**0.2022**	**0.2740**	Sync-MixVoxels	**31.94**	**0.9341**	**0.1266**	**0.2398**
K-Planes hybrid	26.19	0.8757	0.1698	0.2559	K-Planes hybrid	29.36	0.9287	0.1146	0.2135
Sync-K-Planes hybrid	**29.17**	**0.9046**	**0.1259**	**0.2112**	Sync-K-Planes hybrid	**31.23**	**0.9421**	**0.0897**	**0.1798**
K-Planes explicit	25.61	0.8675	0.1891	0.2728	K-Planes explicit	29.39	0.9251	0.1371	0.2350
Sync-K-Planes explicit	**28.56**	**0.9048**	**0.1295**	**0.2114**	Sync-K-Planes explicit	**31.53**	**0.9453**	**0.0869**	**0.1870**
sear_steak					**average**				
MixVoxels	30.88	0.9327	0.1329	0.2453	MixVoxels	29.96	0.9059	0.1669	0.2648
Sync-MixVoxels	**31.27**	**0.9346**	**0.1259**	**0.2415**	Sync-MixVoxels	**30.53**	**0.9101**	**0.1570**	**0.2575**
K-Planes hybrid	31.03	0.9429	0.0925	0.1854	K-Planes hybrid	29.16	0.9120	0.1278	0.2222
Sync-K-Planes hybrid	**31.40**	**0.9453**	**0.0863**	**0.1815**	Sync-K-Planes hybrid	**30.44**	**0.9243**	**0.1064**	**0.1989**
K-Planes explicit	30.66	0.9398	0.1013	**0.2015**	K-Planes explicit	28.51	0.9042	0.1484	0.2438
Sync-K-Planes explicit	**30.71**	**0.9407**	**0.0996**	0.2049	Sync-K-Planes explicit	**29.97**	**0.9223**	**0.1144**	**0.2103**

Table S4: **Per-scene performance in the *test view* on Unsynchronized Plenoptic Video Dataset.** Our method improves all the baselines across all scenes.

train views	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
coffee_martini w/o Camera 13					**coffee_martini w/ Camera 13**				
Mixvoxels	29.47	0.8875	0.1892	0.2733	Mixvoxels	29.54	0.8877	0.1909	0.2738
Sync-Mixvoxels	**29.95**	**0.8915**	**0.1819**	**0.2691**	Sync-Mixvoxels	**30.09**	**0.8921**	**0.1827**	**0.2694**
K-Planes hybrid	29.17	0.8926	0.1690	0.2544	K-Planes hybrid	29.42	0.8928	0.1687	0.2554
Sync-K-Planes hybrid	**31.09**	**0.9138**	**0.1346**	**0.2156**	Sync-K-Planes hybrid	**29.72**	**0.9032**	**0.1466**	**0.2299**
K-Planes explicit	29.14	0.8921	0.1558	0.2465	K-Planes explicit	27.63	0.8625	0.2187	0.3000
Sync-K-Planes explicit	**30.64**	**0.9073**	**0.1334**	**0.2230**	Sync-K-Planes explicit	**29.34**	**0.8986**	**0.1418**	**0.2325**
cook_spinach					**cut_rosted_beef**				
Mixvoxels	32.34	0.9253	0.1410	0.2498	Mixvoxels	32.86	0.9330	0.1289	0.2413
Sync-Mixvoxels	**33.87**	**0.9336**	**0.1235**	**0.2383**	Sync-Mixvoxels	**34.43**	**0.9398**	**0.1132**	**0.2299**
K-Planes hybrid	30.15	0.9220	0.1247	0.2250	K-Planes hybrid	29.83	0.9200	0.1270	0.2302
Sync-K-Planes hybrid	**30.21**	**0.9285**	**0.1016**	**0.1980**	Sync-K-Planes hybrid	**31.38**	**0.9313**	**0.0994**	**0.2000**
K-Planes explicit	29.34	0.9039	0.1584	0.2557	K-Planes explicit	30.23	0.9231	0.1172	0.2162
Sync-K-Planes explicit	**30.38**	**0.9301**	**0.0981**	**0.1942**	Sync-K-Planes explicit	**31.92**	**0.9347**	**0.0946**	**0.1908**
flame_salmon					**flame_steak**				
Mixvoxels	29.36	0.8799	0.1979	0.2834	Mixvoxels	**30.33**	0.9292	0.1313	0.2430
Sync-Mixvoxels	**29.97**	**0.8842**	**0.1899**	**0.2789**	Sync-Mixvoxels	29.94	**0.9300**	**0.1208**	**0.2359**
K-Planes hybrid	26.98	0.8760	0.2031	0.2835	K-Planes hybrid	29.33	0.9227	0.1341	0.2295
Sync-K-Planes hybrid	**31.07**	**0.9131**	**0.1334**	**0.2146**	Sync-K-Planes hybrid	**29.83**	**0.9370**	**0.0933**	**0.1849**
K-Planes explicit	27.35	0.8792	0.1839	0.2779	K-Planes explicit	29.58	0.9237	0.1219	0.2187
Sync-K-Planes explicit	**28.95**	**0.9017**	**0.1400**	**0.2312**	Sync-K-Planes explicit	**30.04**	**0.9385**	**0.0905**	**0.1794**
sear_steak					**average**				
Mixvoxels	34.04	0.9381	0.1159	0.2311	Mixvoxels	31.13	0.9115	0.1565	0.2565
Sync-Mixvoxels	**34.86**	**0.9419**	**0.1076**	**0.2250**	Sync-Mixvoxels	**31.87**	**0.9162**	**0.1457**	**0.2495**
K-Planes hybrid	32.57	0.9402	0.0963	0.1924	K-Planes hybrid	29.64	0.9095	0.1461	0.2386
Sync-K-Planes hybrid	**32.24**	**0.9398**	**0.0941**	**0.1916**	Sync-K-Planes hybrid	**30.79**	**0.9238**	**0.1147**	**0.2050**
K-Planes explicit	31.46	0.9244	0.1275	0.2189	K-Planes explicit	29.25	0.9013	0.1548	0.2477
Sync-K-Planes explicit	**32.87**	**0.9435**	**0.0845**	**0.1786**	Sync-K-Planes explicit	**30.59**	**0.9221**	**0.1118**	**0.2042**

Table S5: **Per-scene performance in the *train view* on Unsynchronized Plenoptic Video Dataset.** Our method enables all baselines to successfully fit unsynchronized training views across all scenes.

test view	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
coffee_martini w/o Camera 13					**coffee_martini w/ Camera 13**				
MixVoxels	**28.97**	**0.8862**	**0.1840**	**0.2681**	MixVoxels	28.79	0.8854	0.1849	0.2724
Sync-MixVoxels	28.76	0.8853	0.1845	0.2698	Sync-MixVoxels	**28.88**	**0.8865**	**0.1817**	**0.2713**
K-Planes hybrid	28.78	0.9034	**0.1282**	**0.2165**	K-Planes hybrid	**28.80**	**0.9025**	**0.1324**	**0.2220**
Sync-K-Planes hybrid	**28.97**	**0.9035**	0.1289	0.2202	Sync-K-Planes hybrid	28.59	0.8994	0.1379	0.2260
K-Planes explicit	28.30	**0.9017**	**0.1375**	**0.2241**	K-Planes explicit	27.78	0.8946	0.1489	0.2384
Sync-K-Planes explicit	**28.39**	0.9016	0.1382	0.2254	Sync-K-Planes explicit	**28.01**	**0.8966**	**0.1464**	**0.2351**
cook_spinach					**cut_rosted_beef**				
MixVoxels	**31.77**	0.9247	0.1423	0.2578	MixVoxels	**31.79**	**0.9253**	0.1401	0.2543
Sync-MixVoxels	31.71	**0.9260**	**0.1399**	**0.2538**	Sync-MixVoxels	31.69	0.9249	**0.1373**	**0.2517**
K-Planes hybrid	31.75	**0.9389**	**0.0901**	0.1896	K-Planes hybrid	**32.26**	**0.9375**	**0.0903**	**0.1937**
Sync-K-Planes hybrid	**32.18**	0.9385	0.0913	**0.1890**	Sync-K-Planes hybrid	31.82	0.9346	0.0936	0.1979
K-Planes explicit	31.10	0.9318	0.1037	0.2043	K-Planes explicit	**31.91**	0.9384	0.0963	0.1991
Sync-K-Planes explicit	**31.24**	**0.9325**	**0.1030**	**0.2034**	Sync-K-Planes explicit	31.88	**0.9386**	**0.0957**	**0.1990**
flame_salmon					**flame_steak**				
MixVoxels	28.79	0.8808	0.1964	0.2718	MixVoxels	31.62	0.9326	0.1290	0.2440
Sync-MixVoxels	**28.93**	**0.8810**	**0.1952**	**0.2692**	Sync-MixVoxels	**31.68**	**0.9341**	**0.1263**	**0.2386**
K-Planes hybrid	28.98	0.9029	0.1285	0.2152	K-Planes hybrid	31.07	**0.9477**	**0.0814**	0.1754
Sync-K-Planes hybrid	**29.15**	**0.9032**	**0.1275**	**0.2138**	Sync-K-Planes hybrid	**31.51**	0.9458	0.0830	**0.1745**
K-Planes explicit	28.63	0.9034	0.1303	0.2139	K-Planes explicit	31.30	0.9446	0.0894	0.1908
Sync-K-Planes explicit	**28.79**	**0.9051**	**0.1277**	**0.2114**	Sync-K-Planes explicit	**31.47**	**0.9451**	**0.0871**	**0.1870**
sear_steak					**average**				
MixVoxels	30.98	**0.9350**	0.1271	0.2419	MixVoxels	30.39	0.9100	0.1577	0.2586
Sync-MixVoxels	**31.24**	0.9349	**0.1262**	**0.2406**	Sync-MixVoxels	**30.41**	**0.9104**	**0.1559**	**0.2564**
K-Planes hybrid	31.16	0.9471	0.0801	0.1739	K-Planes hybrid	30.40	**0.9257**	**0.1044**	**0.1980**
Sync-K-Planes hybrid	**31.70**	**0.9474**	**0.0795**	**0.1771**	Sync-K-Planes hybrid	**30.56**	0.9246	0.1059	0.1998
K-Planes explicit	**31.24**	0.9458	**0.0855**	**0.1874**	K-Planes explicit	30.04	0.9229	0.1131	0.2083
Sync-K-Planes explicit	31.18	**0.9466**	0.0866	0.1893	Sync-K-Planes explicit	**30.14**	**0.9237**	**0.1121**	**0.2072**

Table S6: **Per-scene performance in the *test view* on Synchronized Plenoptic Video Dataset.** Our method improves the baselines even on the synchronized setting by resolving the small temporal errors.

test view	PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$		PSNR	SSIM	LPIPS$_{alex}$	LPIPS$_{vgg}$
box					**deer**				
MixVoxels	27.07	0.9600	0.0474	0.0571	MixVoxels	32.93	0.9652	0.0307	0.0377
Sync-MixVoxels	**30.56**	**0.9715**	**0.0391**	**0.0488**	Sync-MixVoxels	**34.87**	**0.9716**	**0.0266**	**0.0352**
K-Planes hybrid	14.06	0.7408	0.4379	0.4096	K-Planes hybrid	28.90	0.9312	0.0872	0.1061
Sync-K-Planes hybrid	**36.28**	**0.9797**	**0.0262**	**0.0535**	Sync-K-Planes hybrid	**33.00**	**0.9588**	**0.0362**	**0.0430**
K-Planes explicit	17.56	0.8705	0.2149	0.2065	K-Planes explicit	27.60	0.9290	0.0960	0.1285
Sync-K-Planes explicit	**36.00**	**0.9790**	**0.0217**	**0.0443**	Sync-K-Planes explicit	**32.28**	**0.9503**	**0.0461**	**0.0704**
fox					**average**				
MixVoxels	31.85	0.9703	0.0310	0.0452	MixVoxels	30.62	0.9652	0.0364	0.0467
Sync-MixVoxels	**35.77**	**0.9817**	**0.0160**	**0.0317**	Sync-MixVoxels	**33.74**	**0.9749**	**0.0272**	**0.0386**
K-Planes hybrid	22.31	0.8593	0.2744	0.2871	K-Planes hybrid	21.76	0.8438	0.2665	0.2676
Sync-K-Planes hybrid	**33.27**	**0.9716**	**0.2629**	**0.0481**	Sync-K-Planes hybrid	**34.18**	**0.9700**	**0.1084**	**0.0482**
K-Planes explicit	18.90	0.7856	0.3274	0.3402	K-Planes explicit	21.35	0.8617	0.2128	0.2251
Sync-K-Planes explicit	**30.38**	**0.9595**	**0.0581**	**0.0894**	Sync-K-Planes explicit	**32.89**	**0.9629**	**0.0420**	**0.0680**

Table S7: **Per-scene performance in the *test view* on Unsynchronized Dynamic Blender Dataset.** Our method enhances performance on all the baselines.

test view	PSNR	SSIM	$\text{LPIPS}_{\text{alex}}$	$\text{LPIPS}_{\text{vgg}}$		PSNR	SSIM	$\text{LPIPS}_{\text{alex}}$	$\text{LPIPS}_{\text{vgg}}$
box					**deer**				
MixVoxels	**35.99**	0.9811	**0.0244**	0.0374	MixVoxels	35.54	0.9709	**0.0225**	**0.0303**
Sync-MixVoxels	34.53	**0.9813**	0.0245	**0.0344**	Sync-MixVoxels	**35.59**	**0.9735**	0.0240	0.0323
K-Planes hybrid	**36.67**	**0.9810**	0.0275	0.0560	K-Planes hybrid	**33.69**	0.9591	**0.0341**	0.0423
Sync-K-Planes hybrid	36.61	0.9800	**0.0233**	**0.0474**	Sync-K-Planes hybrid	33.53	**0.9594**	0.0346	**0.0419**
K-Planes explicit	35.80	0.9782	0.0324	0.0672	K-Planes explicit	**32.58**	**0.9511**	**0.0470**	**0.0686**
Sync-K-Planes explicit	**36.37**	**0.9797**	**0.0254**	**0.0512**	Sync-K-Planes explicit	32.55	0.9500	0.0513	0.0733
fox					**average**				
MixVoxels	35.96	0.9794	0.0175	0.0337	MixVoxels	**35.83**	0.9771	**0.0214**	0.0338
Sync-MixVoxels	**36.19**	**0.9821**	**0.0155**	**0.0304**	Sync-MixVoxels	35.44	**0.9790**	**0.0214**	**0.0324**
K-Planes hybrid	34.35	0.9722	0.0303	**0.0588**	K-Planes hybrid	34.90	0.9708	0.0306	0.0524
Sync-K-Planes hybrid	**34.96**	**0.9742**	**0.0263**	0.0644	Sync-K-Planes hybrid	**35.03**	**0.9712**	**0.0281**	**0.0512**
K-Planes explicit	**29.49**	**0.9536**	**0.0826**	**0.1162**	K-Planes explicit	32.63	**0.9610**	0.0540	0.0840
Sync-K-Planes explicit	29.17	0.9514	0.0850	0.1188	Sync-K-Planes explicit	**32.70**	0.9604	**0.0539**	**0.0811**

Table S8: **Per-scene performance in the *test view* on Synchronized Dynamic Blender Dataset.** The results of our method in the unsynchronized setting (Table S7) are nearly comparable to the performance in the synchronized setting.

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: 13485–13498.

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*, 5521–5531.

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.