Optimizing Dynamic NeRF and 3DGS with No Video Synchronization

Seoha Kim[1⋆], Jeongmin Bae[1⋆], Youngsik Yun[1], Hyunseung Son[1], Hahyun Lee[2], Gun Bang[2], and Youngjung Uh[1]

[1] Yonsei University, Seoul 03722, Korea
`{jaymin.bae, hailey07, bbangsik, ghfod0917, yj.uh}@yonsei.ac.kr`
[2] Electronics and Telecommunications Research Institute, Daejeon 34129, Korea
`{hanilee, gbang}@etri.re.kr`

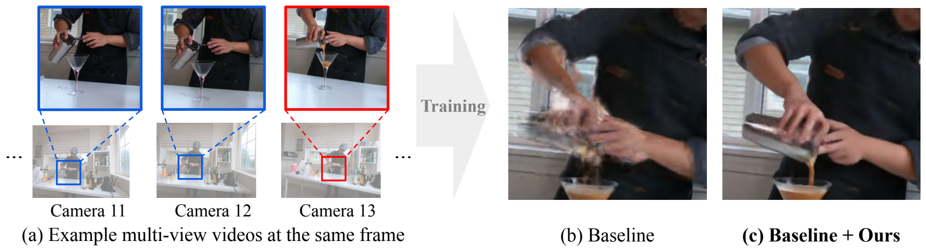

Fig. 1: Teaser. (a) The commonly used Plenoptic Video Dataset in 4D scene reconstruction contains an unsynchronized video. (b) If we include this view in the training set, the baseline fails to reconstruct the motion on the unsynchronized view. (c) In the same settings, ours significantly improves the performance.

Abstract. Recent advancements in 4D scene reconstruction using dynamic NeRF and 3DGS 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 the field. By design, our method is applicable for various baselines, even regardless of the types of radiance fields. We conduct experiments on the common Plenoptic Video Dataset and a newly built Unsynchronized Dynamic Blender Dataset to verify the performance of our method.
Code will be available: https://github.com/seoha-kim/Sync-4DRF

Keywords: Dynamic scene reconstruction · Neural radiance field · Gaussian splatting

⋆ Authors contributed equally to this work.

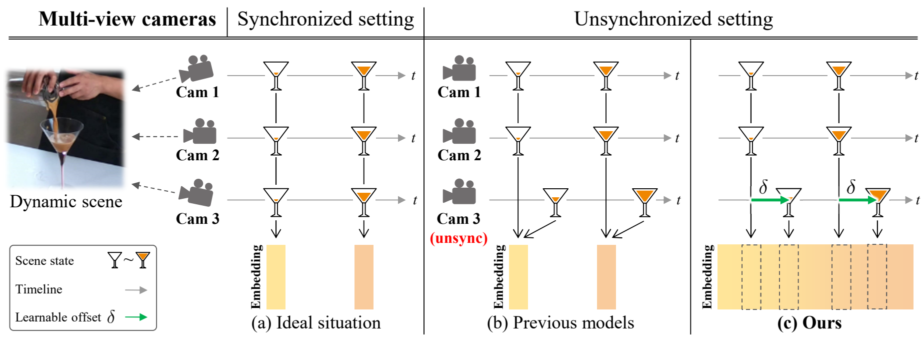

Fig. 2: Problem statement. (a) Ideally, all multi-view images at a frame capture the same moment of a scene. Each frame is represented by a temporal embedding. (b) However, if some frames are not synchronized, previous methods suffer from the discrepancy between the latent embedding of the frame and the actual status of the scene. (c) Our method assigns correct temporal latent embeddings to videos by introducing learnable time offsets δ for individual cameras. The offsets calibrate the time embeddings by shifting them along the time axis.

1 Introduction

Neural radiance fields (NeRF) [27] aims to synthesize novel views given its limited number of views. Since 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 [6, 8, 19, 35, 36] and streamability [1, 18, 32] of dynamic NeRFs.

On the other hand, the newly emerging 3D Gaussian Splatting (3DGS) [13] offers the benefit of real-time rendering by utilizing a GPU-friendly rasterizer for Gaussian primitives. Given its nature as a continuous volumetric radiance field, recent works [39,41] have adopted the approach of establishing a canonical 3DGS and then deforming it to match individual frames, similar to the approach used in deformable NeRFs [30]. However, regardless of the types of radiance fields, it is a common rule that multi-view datasets need to be synchronized.

Our research is motivated by the inaccurate video synchronization in a widely used multi-view dynamic dataset [19]. Although typical synchronization approaches such as timecode systems or audio peaks usually work, these processes have limitations: they require an additional device, cannot be applied in noisy environments, or can be inaccurate.

Figure 1a shows that the rightmost video (red box) is severely ahead of others in the temporal axis. Previous dynamic radiance fields show high-fidelity reconstruction by manually omitting the unsynchronized video, but otherwise, they fail to reconstruct this video in the out-of-sync training view (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.

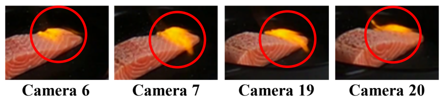

Fig. 3: Temporal discrepancies at the same frame.

The aforementioned problem arises because existing dynamic radiance fields 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 synchronized, there can be a temporal mismatch within a frame (See Figure 3).

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. We emphasize the importance of temporal offset estimation in reconstructing dynamic scenes from in-the-wild settings, analogous to the role of camera pose estimation in static scene reconstruction.

To address this, we introduce time offsets for individual videos and optimize them jointly with dynamic NeRF or 3DGS. The offset 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-4DRF*.

We further design a continuous function that receives time and produces temporal representation to apply our method on dynamic radiance fields with discrete temporal representation [35]. In the case of grid-based approaches, we integrate time offset in bilinear interpolation from spatiotemporal grid representations [4, 8].

Finally, To show the multiple advantages of our method, we design new benchmark datasets by randomly unsynchronizing existing widely used datasets. Since there is no perfect synchronization in the real-world dataset, we newly build an unsynchronized synthetic dataset for the ground truth time offsets.

2 Related Work

Dynamic NeRF and 3DGS. Dynamic NeRFs have been evolving by introducing better representation. D-NeRF models deformation field to extend the static NeRF to dynamic domain [30]. DyNeRF achieves complex temporal representation by implicitly assigning a latent vector to each frame [19]. NeRFPlayer or MixVoxels improve the streamability of dynamic scenes by utilizing grid-based NeRF [19, 35]. K-Planes and HexPlane adopt planar factorization for extending to arbitrary dimensions, enabling the representation of dynamic scenes [4, 8].

After the emergence of 3DGS [13], several studies extend 3DGS to dynamic scene reconstruction. 4DGaussians uses multi-resolution HexPlane [4] to represent temporal deformation of 3DGS [39]. D3DGS employs an implicit function to handle the temporal and spatial changes of the Gaussian [41]. 4DGS decomposes 4D Gaussians into 3D Gaussians conditioned on time and separate 1D marginal Gaussians [40]. STG represents changes in 3D Gaussians over time using polynomial functions [20]. E-D3DGS represent deformation for each Gaussian as the combination of per-Gaussian embeddings and temporal embeddings [2].

All the above methods assume multi-view synchronized video inputs, failing to reconstruct motion in unsynchronized views. We tackle this limitation by introducing learnable time offsets, which can be seamlessly adapted to the existing model. On the other hand, the methods for monocular video settings are relatively free from the synchronization [21,29]. Hence, We do not consider a monocular video setting. Nevertheless, we note that they rely on unnatural teleporting cameras as mentioned in [10] or exhibit holes, artifacts, and unnatural geometry [23,37] compared to those in multi-view video settings.

Embedding for radiance field. Some methods extend NeRF with multiple latent embeddings to represent multiple scenes or different states of a scene. NeRF-W [25] and Block-NeRF [33] use per-image appearance embeddings for different states of a scene to reconstruct a scene from unstructured image collections. Similarly, WildGaussians utilizes per-Gaussian appearance embeddings [16]. ML-NSG employs sequence latent vectors and dynamic latent vectors to capture appearance and geometry variations [7]. LERF and Open3DRF learn language embeddings for encoding open-vocabulary semantic meanings [14,17].

In dynamic radiance fields, D-NeRF represents dynamic scenes using per-frame deformation embedding. Nerfies and HyperNeRF [28,29] apply both per-frame appearance embedding and per-frame deformation embedding. 4DGS represents dynamic scenes using time-conditioned 3D Gaussians and marginal 1D Gaussians [40]. E-D3DGS employs coarse-fine temporal embeddings to model slow or large changes and fast or detailed changes [2].

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, existing dynamic radiance fields can represent dynamic scenes successfully in unsynchronized settings.

Joint camera calibration. NeRF−−, BARF, and SCNeRF [11,11,22] optimize camera parameters and NeRF parameters jointly to eliminate the requirements of known camera parameters in the static setting. RoDyNeRF [24] optimizes dynamic NeRF jointly with camera parameters to tackle the failure of COLMAP in highly dynamic scenes. NoPe-NeRF utilizes monocular depth priors to constrain the estimated relative poses [3]. Similarly in Gaussian Splattings, CF-3DGS optimizes camera parameters jointly with Gaussian splatting [9]. COGS progressively estimates camera poses by utilizing monocular depth and projecting pixels into the 3D world [12]. InstantSplat refines the camera parameters from an off-the-shelf foundation model [5].

Although several methods in static scene reconstruction have jointly learned the camera parameters, none address the problem of inaccurate synchronization across multi-view videos. Traditional approaches for synchronization utilize timecode systems [19] or audio peaks recorded simultaneously with videos [31]. Consequently, these methods require additional devices, and they may not produce accurate results or cannot be applied to videos with significant noise. While Sync-NeRF [15] has demonstrated its effectiveness in dynamic NeRFs, training dynamic 3DGS from unsynchronized videos remains unexplored. We observe that dynamic 3DGS also suffer from quality degradation in these settings, and the proposed time offsets work robustly for Gaussian representations as well.

3 Method

In this section, we explain the joint optimization of per-camera time offsets along with dynamic NeRFs (Section 3.1) and dynamic gaussians (Section 3.2). Subsequently, we describe an implicit function-based approach for models with per-frame temporal embeddings (Section 3.3) and an interpolation-based approach for grid-based models (Section 3.4).

3.1 Sync-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: $t + \delta_k$. 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:

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

We design the time offsets to be continuous rather than discrete frame indices. Further details are deferred to Section 3.3 and 3.4. The time offsets are jointly optimized with NeRF parameters by minimizing MSE between the ground truth RGB pixel $\mathbf{C}$ and volume-rendered RGB pixel $\hat{\mathbf{C}}$:

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

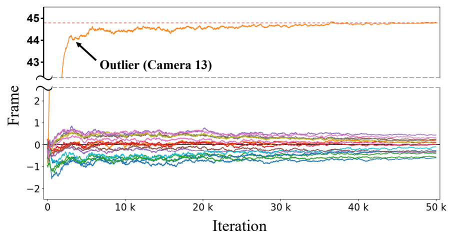

Fig. 4: Learning curve of time offsets. We show camera offsets in `coffee_martini` scene along the training iterations. Our method successfully finds the offset of the outlier camera.

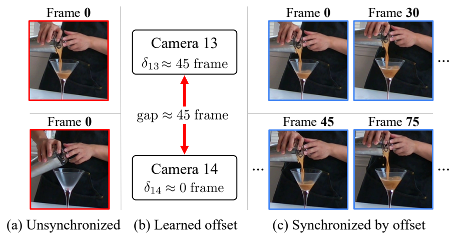

Fig. 5: Synchronization with time offsets. Given unsynchronized videos, our method finds the time offsets, which is equivalent to automatically synchronizing the videos.

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 to approximate volume rendering integral as [26,27]. The time offsets resolve the disagreement across multi-view supervisions and significantly improve the reconstruction quality, especially on the dynamic parts.

Figure 4 is an exemplar plot of the learned time offsets over training iterations by Sinc-MixVoxels on `coffee_martini` scene. 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.

For rendering a video from a novel view, we can customize the time offset δ_{test}. Our method allows us to optimize δ_{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 test-time offset for the test view when we report the test view performance.

3.2 Sync-Gaussians

A recent scene representation, 3DGS, is capable of real-time rendering thanks to a well-tailored GPU-based rasterizer, achieving state-of-the-art visual quality. 3DGS reconstructs the 3D scene by optimizing the set of 3D Gaussians via a differentiable rasterizer. Each 3D Gaussian is characterized by a center $\mathbf{x}_i$, and a covariance Σ defined by rotation R_i and scale S_i:

$$G_i(x) = e^{-\frac{1}{2}(x-\mathbf{x}_i)^T \Sigma_i^{-1}(x-\mathbf{x}_i)}, \qquad \text{where} \quad \Sigma_i = R_i S_i S_i^T R_i^T. \tag{4}$$

To project 3D Gaussian into 2D image space, the 2D covariance Σ' is calculated as follows [43]:

$$\Sigma' = JW\Sigma W^T J^T, \tag{5}$$

where J is the Jacobian of the affine approximation of the projective transformation and W is the world-to-camera transformation.

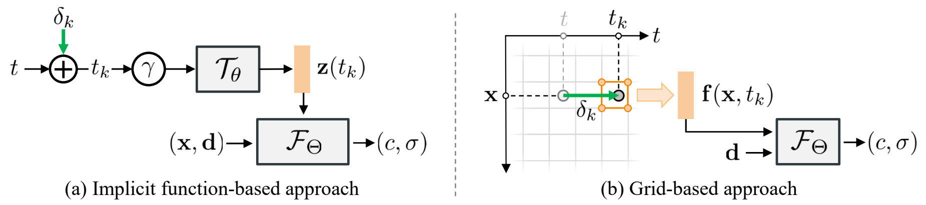

Fig. 6: Continuous temporal embedding. (a) We present an implicit function-based approach for the methods utilizing per-frame temporal embeddings. We add time offset δ_k of camera k to time input t. $\mathcal{T}_\theta$ is the implicit function for mapping calibrated time into temporal embedding $\mathbf{z}$. (b) We query the embedding at the calibrated time t_k on grid-based models. Bilinear interpolation naturally allows continuous temporal embedding.

Each Gaussian has the following learnable parameters: position $\mathbf{x} \in \mathbb{R}^3$, rotation factor $\mathbf{r} \in \mathbb{R}^4$, scaling factor $\mathbf{s} \in \mathbb{R}^3$, spherical harmonic (SH) coefficients Y, and opacity $\sigma \in \mathbb{R}^+$. The rendered pixel color $\hat{\mathbf{C}}$ is calculated by α-blending from $\mathcal{N}$ ordered Gaussians G overlapping the pixel:

$$\hat{\mathbf{C}} = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \qquad \text{where} \quad \alpha_i = \sigma_i G_i(x), \tag{6}$$

and the color c_i is computed using SH coefficients and viewing direction.

To extend 3DGS into 4D scene reconstruction, several works [2,39,41] introduce a deformation function $\mathcal{D}_\Theta$ to predict the deformation of 3D Gaussians for a given timestamp t. The deformation function can be implemented either as an implicit function [41] or as a grid-based representation [39], similar to dynamic NeRF.

As explained in the previous section, by using the calibrated time $t + \delta_k$ as the input for the deformation function, we can predict the rectified variation of Gaussian parameters:

$$\mathcal{D}_\Theta : (G, t + \delta_k) \rightarrow (\Delta\mathbf{x}, \Delta\mathbf{r}, \Delta\mathbf{s}, \Delta Y, \Delta\sigma). \tag{7}$$

After obtaining the set of deformed Gaussians, we can render the scene at time t_k in the same way as 3DGS. The parameters of the Gaussians, the deformation function, and the time offsets for each camera are optimized through a loss function consisting of $\mathcal{L}_1$ and D-SSIM terms: $\mathcal{L}_{\text{RGB}} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$.

3.3 Implicit function for temporal embedding

Various dynamic radiance fields [19, 29, 35] 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 is their interpolation 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 6a). First, we encode a normalized time input t using a set of sinusoidal functions:

$$\gamma(t, L) = \left[\sin(2^0 \pi t), \cdots, \sin(2^{L-1}\pi t), \cos(2^{L-1}\pi t)\right]^{\mathrm{T}}. \tag{8}$$

Similar to previous observations [27, 34], 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:

$$\mathcal{F}_\Theta : (\mathbf{x}, \mathbf{d}, \mathbf{z}(t_k)) \rightarrow (\mathbf{c}, \sigma), \text{ where } \mathbf{z}(t) = \mathcal{T}_\theta(\gamma(t, L)), \tag{9}$$

where t_k is the time input shifted by the per-camera time offset used in Eq. 2. To capture the rapid motions, 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 Section 4.

3.4 Grid-based approaches with time offset

Grid-based dynamic radiance fields [2, 8, 39] 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:

$$\mathcal{F}_\Theta : (\mathbf{f}(\mathbf{x}, t_k), \mathbf{d}) \rightarrow (\mathbf{c}, \sigma), \text{ where } \mathbf{f}(\mathbf{x}, t) = \mathrm{Grid}(\mathbf{x}, t), \tag{10}$$

and $\mathrm{Grid}(\cdot)$ denotes the interpolation of grid vectors surrounding given coordinates. Figure 6b illustrates how our method modifies the sampling in the grid by δ_k. We select K-Planes and 4DGaussians as a baseline for using grid-based representation and compare it with our Sync-K-Planes and Sync-4DGaussians.

4 Experiments

In this section, we validate the effectiveness of our method. 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.

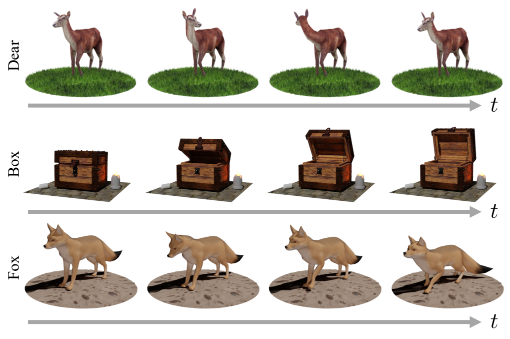

Fig. 7: Snapshots of Unsynchronized Dynamic Blender Dataset.

4.1 Experiment settings

Unsynchronized datasets. The Plenoptic Video Dataset [19] contains six challenging real-world scenes with varying degrees of dynamics. Its multi-view videos are roughly synchronized except `coffee_martini` scene. To simulate an in-the-wild unsynchronized capturing environment, we modify this dataset to be unsynchronized 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.

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 7. The dataset is publicly available.

Baselines. We employ MixVoxels, K-Planes with hybrid encoder, and 4DGaussians as our baselines and adopt our method upon them, namely, Sync-MixVoxels, Sync-K-Planes, and Sync-4DGaussians. 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 [38]. To measure higher-level perceptual similarity, we report LPIPS [42] using VGG and AlexNet. Higher

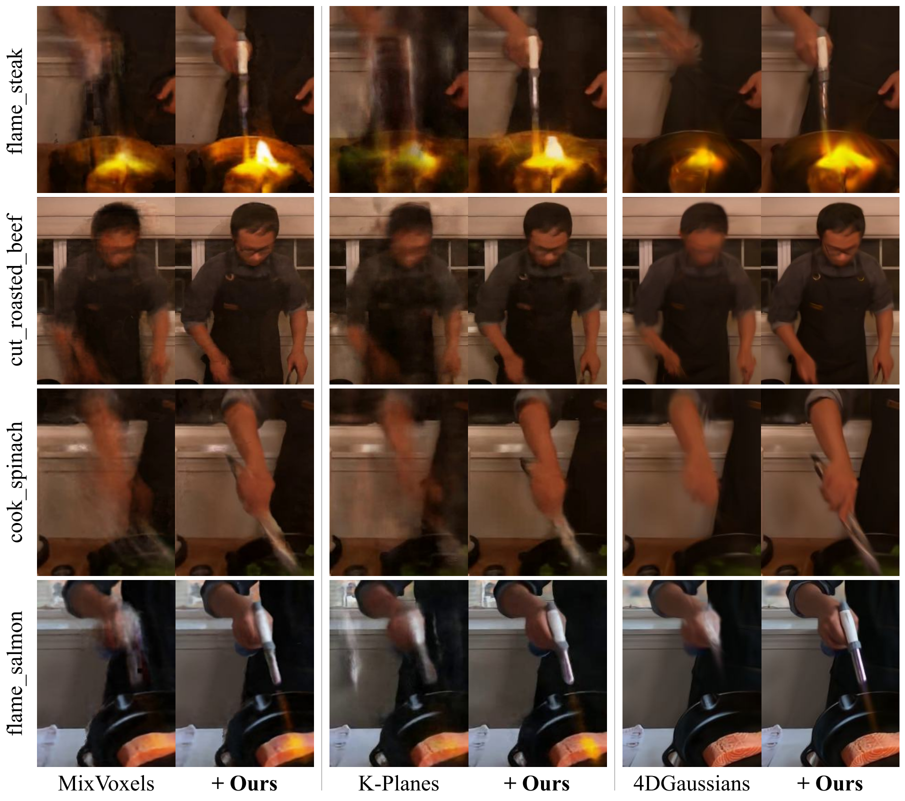

Fig. 8: Cropped renderings on Unsynchronized Plenoptic Video Dataset.

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

Unsynchronized Plenoptic Video Dataset. Figure 8 compares ours with the baselines on the Unsynchronized Plenoptic Video Dataset. All the baselines produce severe artifacts on dynamic parts and suffer worse on 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.

Above all, our method successfully corrects artifacts in all baselines. Additionally, in Figure 9, 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

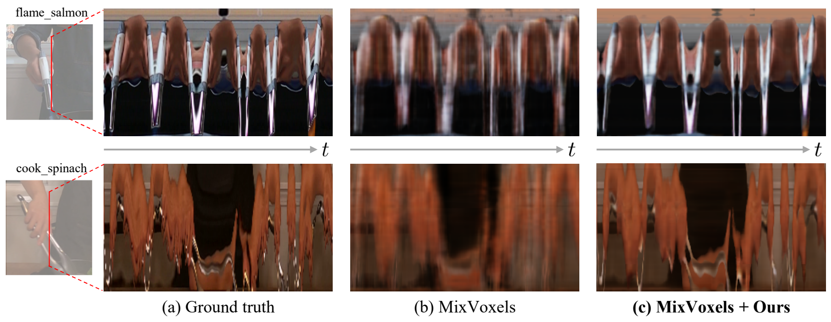

Fig. 9: Spatiotemporal images. Each column in the image represents the rendered pixels on the fixed vertical line at a moment. The fixed vertical line with 150 pixels is shown in the leftmost image patch. We render all frames in the video, producing 270 pixel wide spatiotemporal images. Our results are much clearer than the baseline.

model \ metric	PSNR	SSIM	LPIPS$_{\text{alex}}$	LPIPS$_{\text{vgg}}$
MixVoxels	29.96	0.9059	0.1669	0.2648
Sync-MixVoxels	**30.53**	**0.9101**	**0.1570**	**0.2575**
K-Planes	29.16	0.9120	0.1278	0.2222
Sync-K-Planes	**30.44**	**0.9243**	**0.1064**	**0.1989**
4DGaussians	29.86	0.9269	0.0683	0.1468
Sync-4DGaussians	**30.51**	**0.9312**	**0.0616**	**0.1412**

Table 1: Average performance on Unsynchronized Plenoptic Video Dataset. Our method improves all the baselines, even achieving performance similar to synchronized setting in Table 5.

from the test view constructs a spatiotemporal image as a whole. Compared to the baseline, our method exhibits significantly clearer spatialtemporal images (Figure 9b-c).

Table 1 reports quantitative metrics on the test views of the unsynchronized Plenoptic Video Dataset. 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.

Unsynchronized Dynamic Blender Dataset. Figure 10 compares the baselines and ours on `fox` and `box` scene from Unsynchronized Dynamic Blender Dataset. While all baselines struggle on the motion and object boundaries, our approaches show clear results of fox and box.

Table 2 reports quantitative metrics in the test view of the Unsynchronized Dynamic Blender Dataset. Our method improves the baselines in all cases.

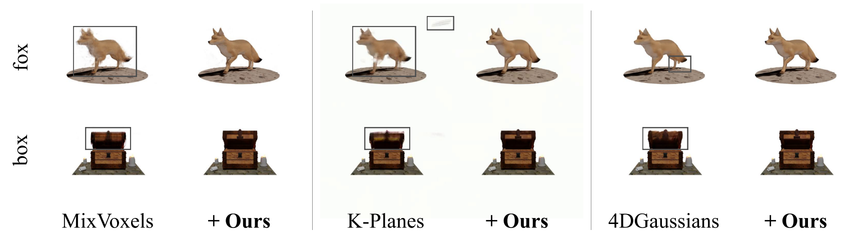

Fig. 10: Qualitative results on Unsynchronized Dynamic Blender Dataset

model \ metric	PSNR	SSIM	LPIPS$_{\text{alex}}$	LPIPS$_{\text{vgg}}$
MixVoxels	31.06	0.9753	0.0237	0.0339
Sync-MixVoxels	**37.11**	**0.9841**	**0.0120**	**0.0226**
K-Planes	32.66	0.9771	0.0175	0.0244
Sync-K-Planes	**39.40**	**0.9863**	**0.0066**	**0.0131**
4DGaussians	34.80	0.9820	0.0123	0.0178
Sync-4DGaussians	**40.64**	**0.9884**	**0.0052**	**0.0106**

Table 2: Average performance on Unsynchronized Dynamic Blender Dataset. Our method improves all the baselines.

4.3 Time offset accuracy

We demonstrate that the time offsets found by our method are highly accurate. Table 3 reports mean absolute error (MAE) of the optimized time offsets between the ground truth on the Unsynchronized Dynamic Blender Dataset. The average error is approximately 0.01 seconds, which corresponds to less than one-third of a frame.

	MAE (seconds)
Sync-MixVoxels	0.0154
Sync-K-Planes	0.0156
Sync-4DGaussians	0.0068

Table 3: MAE between predicted offsets and ground truth offsets. 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 in the test view under different lengths of unsynchronization, namely 1.5× and 2× long offsets. Table 4 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.

model \ scene	1.5× cook spinach	1.5× flame salmon	1.5× fox	2.0× cook spinach	2.0× flame salmon	2.0× fox
MixVoxels	30.21	27.27	31.10	30.48	27.66	29.81
Sync-MixVoxels	**31.44**	**28.59**	**36.15**	**31.50**	**28.74**	**35.58**
K-Planes	29.93	26.11	31.85	29.13	26.01	29.71
Sync-K-Planes	**31.71**	**28.67**	**40.31**	**31.85**	**28.22**	**40.36**
4DGaussians	31.15	27.73	33.78	30.73	27.53	31.31
Sync-4DGaussians	**31.77**	**28.58**	**36.73**	**31.64**	**28.09**	**38.80**

Table 4: Average PSNR with varaious unsynchonization lengths. Our method is robust to different lengths of unsynchronization.

model \ metric	PSNR	SSIM	LPIPSalex	LPIPSvgg
MixVoxels	30.39	0.9100	0.1577	0.2586
Sync-MixVoxels	**30.41**	**0.9104**	**0.1559**	**0.2564**
K-Planes	30.40	**0.9257**	**0.1044**	**0.1980**
Sync-K-Planes	**30.56**	0.9246	0.1059	0.1998
4DGaussians	30.15	0.9275	0.0657	0.1596
Sync-4DGaussians	**30.47**	**0.9271**	**0.0660**	**0.1598**

Table 5: Average performance on *Synchronized* Plenoptic Video Dataset. Our method enhances performance on synchronized setting by resolving the small temporal gaps.

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 the 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.

5 Ablation and tuning

5.1 Optimizing test view offset

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 on Sync-MixVoxels and 1K iterations on Sync-K-Planes and Sync-4DGaussians, and then perform the evaluation. We train only the time offset while freezing the entire network. Table 6 shows that optimizing the time offset in the test view improves performance slightly.

5.2 Sinusoidal function for temporal embedding

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 7 show the results of Sync-MixVoxels on `flame_salmon` scene for different Ls.

average	PSNR	SSIM	LPIPS$_{\text{alex}}$	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	**30.44**	**0.9243**	**0.1064**	**0.1989**
⌞w/o δ_{test}	30.25	0.9235	0.1067	0.1992
Sync-4DGaussians	**30.51**	**0.9312**	**0.0616**	**0.1412**
⌞w/o δ_{test}	30.33	0.9300	0.0619	0.1416

Table 6: Comparison with test view offset optimization on Unsynchronized Plenoptic Video Dataset. Test view offset optimization enables more accurate evaluation and improves our quantitative results.

L for time	PSNR	SSIM	LPIPS$_{\text{alex}}$	LPIPS$_{\text{vgg}}$
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 7: 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.

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.

6 Conclusion

Our work is the first attempt to train dynamic radiance fields on unsynchronized multi-view videos. We have shown that the existing dynamic NeRF and 3DGS 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 radiance field with typical reconstruction loss. Our method, Sync-4DRF, is versatile to the types of fields, dynamic NeRF or 3DGS.

Discussion Implicit function for temporal embeddings requires 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. Meanwhile, our method applied to grid-based temporal embedding does not introduce additional computational complexity.

	Rendering	Training	# Params
MixVoxels	2.3 min	1.9 hrs	130.7 M
Sync-MixVoxels	2.5 min	2.5 hrs	131.8 M
K-Planes	15.0 min	1.9 hrs	27.0 M
Sync-K-Planes	15.2 min	2.0 hrs	27.0 M
4DGaussians	12.2 sec	47.9 min	12.9 M
Sync-4DGaussians	13.2 sec	45.8 min	13.7 M

Table 8: Computational cost in the `flame_salmon` scene.

Acknowledgements

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

1. Attal, B., Huang, J.B., Richardt, C., Zollhoefer, M., Kopf, J., O'Toole, M., Kim, C.: Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16610–16620 (2023) 2
2. Bae, J., Kim, S., Yun, Y., Lee, H., Bang, G., Uh, Y.: Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In: European Conference on Computer Vision (ECCV) (2024) 4, 7, 8
3. Bian, W., Wang, Z., Li, K., Bian, J.W., Prisacariu, V.A.: Nope-nerf: Optimising neural radiance field with no pose prior. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4160–4169 (June 2023) 4
4. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 130–141 (2023) 3
5. Fan, Z., Cong, W., Wen, K., Wang, K., Zhang, J., Ding, X., Xu, D., Ivanovic, B., Pavone, M., Pavlakos, G., Wang, Z., Wang, Y.: Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds (2024) 4
6. Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast dynamic radiance fields with time-aware neural voxels. In: SIGGRAPH Asia 2022 Conference Papers (2022) 2
7. Fischer, T., Porzi, L., Bulò, S.R., Pollefeys, M., Kontschieder, P.: Multi-level neural scene graphs for dynamic urban environments (2024), https://arxiv.org/abs/2404.00168 4
8. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023) 2, 3, 8
9. Fu, Y., Liu, S., Kulkarni, A., Kautz, J., Efros, A.A., Wang, X.: Colmap-free 3d gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20796–20805 (June 2024) 4
10. Gao, H., Li, R., Tulsiani, S., Russell, B., Kanazawa, A.: Monocular dynamic view synthesis: A reality check. In: NeurIPS (2022) 4
11. Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields. In: Proceedings of the Int. Conf. on Computer Vision (ICCV) (2021) 4
12. Jiang, K., Fu, Y., Varma T, M., Belhe, Y., Wang, X., Su, H., Ramamoorthi, R.: A construct-optimize approach to sparse view synthesis without camera pose. SIGGRAPH (2024) 4
13. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023), https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/ 2, 3

14. Kerr, J., Kim, C.M., Goldberg, K., Kanazawa, A., Tancik, M.: Lerf: Language embedded radiance fields. In: International Conference on Computer Vision (ICCV) (2023) 4
15. Kim, S., Bae, J., Yun, Y., Lee, H., Bang, G., Uh, Y.: Sync-nerf: Generalizing dynamic nerfs to unsynchronized videos. Proceedings of the AAAI Conference on Artificial Intelligence **38**, 2777–2785 (03 2024). https://doi.org/10.1609/aaai.v38i3.28057 5
16. Kulhanek, J., Peng, S., Kukelova, Z., Pollefeys, M., Sattler, T.: Wildgaussians: 3d gaussian splatting in the wild (2024), https://arxiv.org/abs/2407.08447 4
17. Lee, H., Yun, Y., Bae, J., Kim, S., Uh, Y.: Rethinking open-vocabulary segmentation of radiance fields in 3d space (2024), https://arxiv.org/abs/2408.07416 4
18. Li, L., Shen, Z., Wang, Z., Shen, L., Tan, P.: Streaming radiance fields for 3d video synthesis. Advances in Neural Information Processing Systems **35** (2022) 2
19. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 2, 3, 5, 7, 9
20. Li, Z., Chen, Z., Li, Z., Xu, Y.: Spacetime gaussian feature splatting for real-time dynamic view synthesis. arXiv preprint arXiv:2312.16812 (2023) 4
21. Li, Z., Wang, Q., Cole, F., Tucker, R., Snavely, N.: Dynibar: Neural dynamic image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023) 4
22. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: IEEE International Conference on Computer Vision (ICCV) (2021) 4
23. Liu, Q., Liu, Y., Wang, J., Lv, X., Wang, P., Wang, W., Hou, J.: Modgs: Dynamic gaussian splatting from causually-captured monocular videos (2024), https://arxiv.org/abs/2406.00434 4
24. Liu, Y.L., Gao, C., Meuleman, A., Tseng, H.Y., Saraf, A., Kim, C., Chuang, Y.Y., Kopf, J., Huang, J.B.: Robust dynamic radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023) 4
25. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In: CVPR (2021) 4
26. Max, N.: Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics **1**(2), 99–108 (1995) 6
27. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021) 2, 6, 8
28. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. ICCV (2021) 4
29. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021) 4, 7
30. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. arXiv preprint arXiv:2011.13961 (2020) 2, 3
31. Shrstha, P., Barbieri, M., Weda, H.: Synchronization of multi-camera video recordings based on audio. In: Proceedings of the 15th ACM international conference on Multimedia. pp. 545–548 (2007) 5

32. Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., Geiger, A.: Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. IEEE Transactions on Visualization and Computer Graphics **29**(5), 2732–2742 (2023) 2
33. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8248–8258 (June 2022) 4
34. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems **33**, 7537–7547 (2020) 8
35. Wang, F., Tan, S., Li, X., Tian, Z., Liu, H.: Mixed neural voxels for fast multi-view video synthesis. arXiv preprint arXiv:2212.00190 (2022) 2, 3, 7
36. Wang, L., Zhang, J., Liu, X., Zhao, F., Zhang, Y., Zhang, Y., Wu, M., Yu, J., Xu, L.: Fourier plenoctrees for dynamic radiance field rendering in real-time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13524–13534 (June 2022) 2
37. Wang, Q., Ye, V., Gao, H., Austin, J., Li, Z., Kanazawa, A.: Shape of motion: 4d reconstruction from a single video (2024), `https://arxiv.org/abs/2407.13764` 4
38. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004) 9
39. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Xinggang, W.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023) 2, 3, 7, 8
40. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: International Conference on Learning Representations (ICLR) (2024) 3, 4
41. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. arXiv preprint arXiv:2309.13101 (2023) 2, 3, 7
42. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018) 9
43. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 371–378 (2001) 6