3D Moments from Near-Duplicate Photos

Qianqian Wang[1,2] Zhengqi Li[1] David Salesin[1] Noah Snavely[1,2] Brian Curless[1,3] Janne Kontkanen[1]
[1]Google Research [2]Cornell Tech, Cornell University [3]University of Washington

Near-duplicate photos

Space-time videos

Figure 1. People often take many near-duplicate photos in an attempt to capture the perfect expression. Given a pair of these photos, taken from nearby viewpoints (**left**), our proposed approach brings these photos to life as 3D Moments, producing space-time videos with cinematic camera motions and interpolated scene motion (**right**). Please refer to the supplementary material to see the videos.

Abstract

We introduce 3D Moments*, a new computational photography effect. As input we take a pair of near-duplicate photos, i.e., photos of moving subjects from similar viewpoints, common in people's photo collections. As output, we produce a video that smoothly interpolates the scene motion from the first photo to the second, while also producing camera motion with parallax that gives a heightened sense of 3D. To achieve this effect, we represent the scene as a pair of feature-based layered depth images augmented with scene flow. This representation enables motion interpolation along with independent control of the camera viewpoint. Our system produces photorealistic space-time videos with motion parallax and scene dynamics, while plausibly recovering regions occluded in the original views. We conduct extensive experiments demonstrating superior performance over baselines on public datasets and in-the-wild photos. Project page:* https://3d-moments.github.io/.

1. Introduction

Digital photography enables us to take scores of photos in order to capture just the right moment. In fact, we often end up with many near-duplicate photos in our image collections as we try to capture the best facial expression of a family member, or the most memorable part of an action. These near-duplicate photos end up just lying around in digital storage, unviewed.

In this paper, we aim to utilize such near-duplicate photos to create a compelling new kind of 3D photo enlivened with animation. We call this new effect *3D Moments*: given a pair of near-duplicate photos depicting a dynamic scene from nearby (perhaps indistinguishable) viewpoints, such as the images in Fig. 1 (left), our goal is to simultaneously enable cinematic camera motion with 3D parallax (including novel, extrapolated viewpoints) while faithfully interpolating scene motion to synthesize short space-time videos like the one shown in Fig. 1 (right). 3D Moments combine both camera and scene motion in a compelling way, but involve very challenging vision problems: we must jointly infer 3D geometry, scene dynamics, and content that becomes newly disoccluded during the animation.

Despite great progress towards each of these individual problems, tackling all of them jointly is non-trivial, especially with image pairs with unknown camera poses as input. NeRF-based view synthesis methods for dynamic scenes [15, 27, 28, 49] require many images with known camera poses. Single-photo view synthesis methods (sometimes called 3D Photos or 3D Ken Burns [11, 25, 38]) can create

animated camera paths from a single photo, but cannot represent moving people or objects. Frame interpolation can create smooth animations from image pairs, but only in 2D. Furthermore, naively applying view synthesis and frame interpolation methods sequentially results in temporally inconsistent, unrealistic animations.

To address these challenges, we propose a novel approach for creating 3D Moments by explicitly modeling time-varying geometry and appearance from two uncalibrated, near-duplicate photos. The key to our approach is to represent the scene as a pair of feature-based layered depth images (LDIs) augmented with scene flows. We build this representation by first transforming the input photos into a pair of color LDIs, with inpainted color and depth for occluded regions. We then extract features for each layer with a neural network to create the feature LDIs. In addition, we compute optical flow between the input images and combine it with the depth layers to estimate scene flow between the LDIs. To render a novel view at a novel time, we lift these feature LDIs into a pair of 3D point clouds, and employ a depth-aware, bidirectional splatting and rendering module that combines the splatted features from both directions.

We extensively test our method on both public multi-view dynamic scene datasets and in-the-wild photos in terms of rendering quality, and demonstrate superior performance compared to state-of-the-art baselines.

In summary, our main contributions include: (1) the new task of creating 3D Moments from near-duplicate photos of dynamic scenes, and (2) a new representation based on feature LDIs augmented with scene flows, and a model that can be trained for creating 3D Moments.

2. Related work

Our work builds on methods for few-shot view synthesis, frame interpolation and space-time view synthesis.

View synthesis from one or two views. Novel view synthesis aims to reconstruct unseen viewpoints from a set of input 2D images. Recent neural rendering methods achieve impressive synthesis results [17, 20, 43, 44, 47, 54], but typically assume many views as input and thus do not suit our task. We focus here on methods that take just one or two views. Many single-view synthesis methods involve estimating dense monocular depths and filling in occluded regions [7, 11, 14, 25, 34, 38, 48], while others seek to directly regress to a scene representation in a single step [30, 35, 45, 46, 53]. We draw on ideas from several works in this vein: SynSin learns a feature 3D point cloud for each input image and projects it to the target view where the missing regions are inpainted [48]. 3D Photo [38] instead creates a Layered Depth Image (LDI) and inpaints the color and depth of the occluded region in a spatial context-aware manner. We build on both methods but extend to the case of dynamic scenes.

Like our method, some prior view synthesis methods operate on two views. For instance, Stereo Magnification [56] and related work [40] take two narrow-baseline stereo images and predict a multi-plane image that enables real-time novel view synthesis. However, unlike our approach, these methods assume that there is some parallax from camera motion, and again can only model static scenes, not ones where there is scene motion between the two input views.

Frame interpolation. In contrast to 3D view synthesis, temporal frame interpolation creates sequences of in-between frames from two input images. Frame interpolation methods do not distinguish between camera and scene motion: all object motions are interpolated in 2D image space. Moreover, most frame interpolators assume a linear motion model [2, 6, 8, 12, 21–24, 26, 39] although some recent works consider quadratic motion [18, 50]. Most of the interpolators use image warping with optical flow, although as a notable exception, Niklaus et al. [23, 24] synthesize intermediate frames by blending the inputs with kernels predicted by a neural network. However, frame interpolation alone cannot generate 3D Moments, since it does not recover the 3D geometry or allow control over camera motion in 3D.

Space-time view synthesis. A number of methods have sought to synthesize novel views for dynamic scenes in both space and time by modeling time-varying 3D geometry and appearance. Many methods require synchronized multi-view videos as inputs, and thus do not apply to in-the-wild photos [1, 3, 4, 13, 41, 57]. Recently, several neural rendering approaches [15, 27–29, 49, 52] have shown promising results on space-time view synthesis from monocular dynamic videos. To interpolate both viewpoints and time, recent works either directly interpolate learned latent codes [27, 28], or apply splatting with estimated 3D scene flow fields [15]. However, these methods require densely sampled input views with accurate camera poses, which are unavailable for our two-image setting. Moreover, none of them explicitly inpaint unseen regions.

3. Method

3.1. Problem statement and method overview

The input to our system is a pair of images $(\mathbf{I}_0, \mathbf{I}_1)$ of a dynamic scene taken at nearby times and camera viewpoints. For tractable motion interpolation, we assume that motion between $\mathbf{I}_0$ and $\mathbf{I}_1$ is roughly within the operating range of a modern optical flow estimator. Our goal is to create 3D Moments by independently controlling the camera viewpoint while simultaneously interpolating scene motion to render arbitrary nearby novel views at arbitrary intermediate times $t \in [0, 1]$. Our output is a space-time video with cinematic camera motions and interpolated scene motion.

To this end, we propose a new framework that enables efficient and photorealistic space-time novel view synthesis

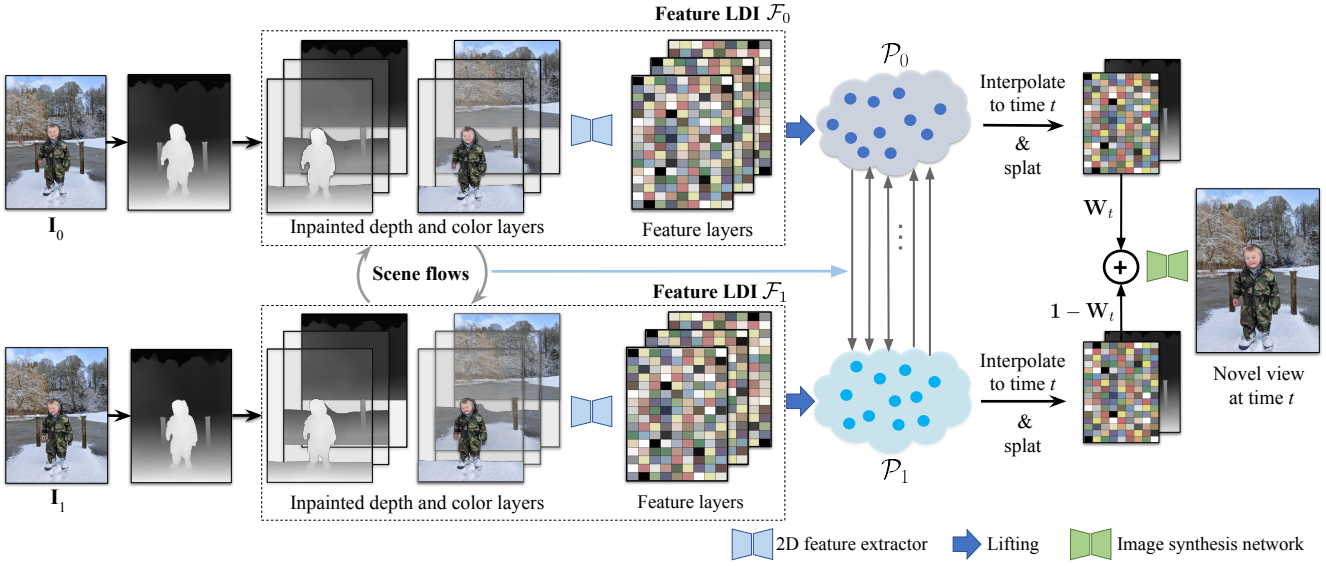


Figure 2. **Overview.** Given near-duplicate photos ($\mathbf{I}_0, \mathbf{I}_1$), we align them with a homography and predict a dense depth map for each photo. Each RGBD image is then converted to a color LDI, with occluded regions filled by depth-aware inpainting. A 2D feature extractor is applied to each color layer of the inpainted LDIs to obtain feature layers, resulting in feature LDIs ($\mathcal{F}_0, \mathcal{F}_1$), where colors in the inpainted LDIs have been replaced with features. To model scene motion, we compute the scene flow of each pixel in the LDIs using the predicted depths and optical flows between the two input images. To render a novel view at intermediate time t, we lift the feature LDIs to a pair of 3D point clouds ($\mathcal{P}_0, \mathcal{P}_1$) and bidirectionally move points along their scene flows to time t. We then project and splat these 3D feature points to form forward and backward 2D feature maps (from $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively) and their corresponding depth maps. We linearly blend these maps with weight map $\mathbf{W}_t$ derived from spatio-temporal cues, and pass the result to an image synthesis network to produce the final image.

without the need for test-time optimization. Our pipeline is illustrated in Fig. 2. Our system starts by aligning the two photos into a single reference frame via a homography. The key to our approach is building feature LDI from each of the inputs, where each pixel in the feature LDI consists of its depth, scene flow and a learnable feature.

To do so, we first transform each input image into a color LDI [37] with inpainted color and depth in occluded regions. We then extract deep feature maps from each color layer of these LDIs to obtain a pair of feature LDIs ($\mathcal{F}_0, \mathcal{F}_1$). To model scene dynamics, the scene flows of each pixel in the LDIs are estimated based on predicted depth and optical flows between the two inputs. Finally, to render a novel view at intermediate time t, we lift the feature LDIs into a pair of point clouds $(\mathcal{P}_0, \mathcal{P}_1)$ and propose a scene-flow-based bidirectional splatting and rendering module to combine the features from two directions and synthesize the final image. We now describe our method in more detail.

3.2. LDIs from near-duplicate photos

Our method first computes the underlying 3D scene geometry. As near-duplicates typically have scene dynamics and very little camera motion, standard Structure from Motion (SfM) and stereo reconstruction methods fail to produce reliable results. Instead, we found that state-of-the-art monocular depth estimator DPT [31] can produce sharp and plausible dense depth maps for images in the wild. Therefore, we rely on DPT to obtain the geometry for each image.

To account for small camera pose changes between the views, we compute optical flow between the views using RAFT [42], estimate a homography between the images using the flow, and then warp $\mathbf{I}_1$ to align with $\mathbf{I}_0$. Because we only want to align the static background of two images, we mask out regions with large optical flow, which often correspond to moving objects, and compute the homography using the remaining mutual correspondences given by the flow. Once $\mathbf{I}_1$ is warped to align with $\mathbf{I}_0$, we treat their camera poses as identical. To simplify notation, we henceforth re-use $\mathbf{I}_0$ and $\mathbf{I}_1$ to denote the aligned input images.

We then apply DPT [32] to predict the depth maps for each image. To align the depth range of $\mathbf{I}_1$ with $\mathbf{I}_0$ we estimate a global scale and shift for $\mathbf{I}_1$'s disparities (i.e., 1/depth), using flow correspondences in the static regions. Next, we convert the aligned photos and their dense depths to an LDI representation [37], in which layers are separated according to depth discontinuities, and apply RGBD inpainting in occluded regions as described below.

Prior methods for 3D photos iterate over all depth edges in an LDI to adaptively inpaint local regions using background pixels of the edge [11,38]. However, we found this procedure

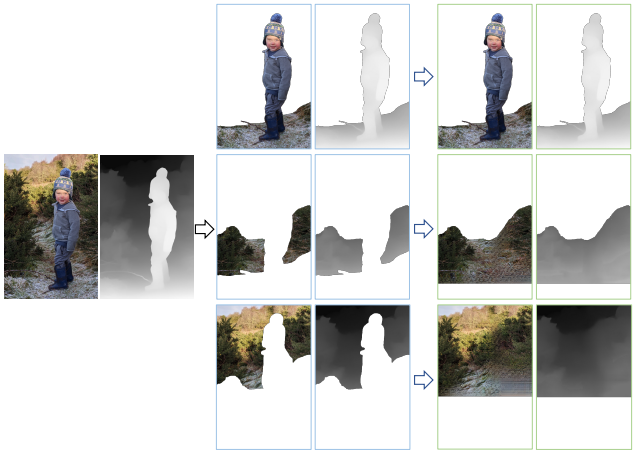
to be computationally expensive and the output difficult to feed into a training pipeline. More recently, Jampani et al. [7] employ a two-layer approach that would otherwise suit our requirements but is restricted in the number of layers. We therefore propose a simple, yet effective strategy for creating and inpainting LDIs that flow well into our learning-based pipeline. Specifically, we first perform agglomerative clustering [19] in disparity space to separate the RGBD maps into different depth layers (Fig. 3 (a)). We set a fixed distance threshold above which clusters will not be merged, resulting in $2 \sim 5$ layers for an image. We apply the clustering to the disparities of both images to obtain their LDIs, $\mathcal{L}_0 \triangleq \{\mathbf{C}_0^l, \mathbf{D}_0^l\}_{l=1}^{L_0}$ and $\mathcal{L}_1 \triangleq \{\mathbf{C}_1^l, \mathbf{D}_1^l\}_{l=1}^{L_1}$, where $\mathbf{C}^l$ and $\mathbf{D}^l$ represent the l^{th} color and depth layer respectively, and L_0 and L_1 denote the number of layers constructed from $\mathbf{I}_0$ and $\mathbf{I}_1$, respectively. Each color layer is an RGBA image, with the alpha channel indicating valid pixels in this layer.

Next, we apply depth-aware inpainting to each color and depth LDI layer in occluded regions. To inpaint missing contents in layer l, we treat all the pixels between the l^{th} layer and the farthest layer as the context region (i.e., the region used as reference for inpainting), and exclude all irrelevant foreground pixels in layers nearer than layer l. We set the rest of the l^{th} layer within a certain margin from existing pixels (see supplement) to be inpainted. We keep only inpainted pixels whose depths are smaller than the maximum depth of layer l so that inpainted regions do not mistakenly occlude layers farther than layer l. We adopt the pre-trained inpainting network from Shih *et al.* [38] to inpaint color and depth at each layer. Fig. 3 (b) shows an example of LDI layers after inpainting. Note that we choose to inpaint the two LDIs up front rather than performing per-frame inpainting for each rendered novel view, as the latter would suffer from multi-view inconsistency due to the lack of a global representation for disoccluded regions.

3.3. Space-time scene representation

We now have inpainted color LDIs $\mathcal{L}_0$ and $\mathcal{L}_1$ for novel view synthesis. From each individual LDI, we could synthesize new views of the static scene. However, the LDIs alone do not model the scene motion between the two photos. To enable motion interpolation, we estimate 3D motion fields between the images. To do so, we first compute 2D optical flow between the two aligned images and perform a forward and backward consistency check to identify pixels with mutual correspondences. Given 2D mutual correspondences, we use their associated depth values to compute their 3D locations and lift the 2D optical flow to 3D scene flow, i.e., 3D translation vectors that displace each 3D point from one time to another. This process gives the scene flow for mutually visible pixels of the LDIs.

However, for pixels that do not have mutual correspondences, such as those occluded in the other view or those



(a) LDI (b) Inpainted LDI

Figure 3. **From an image to an inpainted LDI.** Given an input image and its estimated monocular depth [31], we first apply agglomerative clustering [19] to separate the RGBD image into multiple (in this example 3) RGBDA layers as shown in (**a**), then perform context-aware color and depth inpainting [38] to obtain inpainted RGBDA layers (**b**).

in the inpainted region, 3D correspondences are not well defined. To handle this issue, we leverage the fact that the scene flows are spatially smooth and propagate them from well-defined pixels to missing regions. In particular, for each pixel in $\mathcal{L}_0$ with a corresponding point in $\mathcal{L}_1$, we store its associated scene flow at its pixel location, resulting in scene flow layers initially containing only well-defined values for mutually visible pixels. To inpaint the remaining scene flow, we perform a diffusion operation that iteratively applies a masked blur filter to each scene flow layer until all pixels in $\mathcal{L}_0$ have scene flow vectors. We apply the same method to $\mathcal{L}_1$ to obtain complete scene flow layers for the second LDI. This process gives us complete forward and backward scene flows for every pixel in $\mathcal{L}_0$ and $\mathcal{L}_1$, respectively.

To render an image from a novel camera viewpoint and time with these two scene-flow-augmented LDIs, one simple approach is to directly interpolate the LDI point locations to the target time according to their scene flow and splat RGB values to the target view. However, when using this method, we found that any small error in depth or scene flow can lead to noticeable artifacts. We therefore correct for such errors by training a 2D feature extraction network that takes each inpainted LDI color layer $\mathbf{C}^l$ as input and produces a corresponding 2D feature map $\mathbf{F}^l$. These features encode local appearance of the scene and are trained to mitigate rendering artifacts introduced by inaccurate depth or scene flow and to improve overall rendering quality. This step converts our inpainted color LDIs to feature LDIs $\mathcal{F}_0 \triangleq \{\mathbf{F}_0^l, \mathbf{D}_0^l\}_{l=1}^{L_0}$, $\mathcal{F}_1 \triangleq \{\mathbf{F}_1^l, \mathbf{D}_1^l\}_{l=1}^{L_1}$, both of which are augmented with scene flows. Finally, we lift all valid pixels for these fea-

ture LDIs into a pair of point clouds $\mathcal{P}_0 \triangleq \{(\mathbf{x}_0, \mathbf{f}_0, \mathbf{u}_0)\}$ and $\mathcal{P}_1 \triangleq \{(\mathbf{x}_1, \mathbf{f}_1, \mathbf{u}_1)\}$, where each point is defined with 3D location $\mathbf{x}$, appearance feature $\mathbf{f}$, and 3D scene flow $\mathbf{u}$.

3.4. Bidirectional splatting and rendering

Given a pair of 3D feature point clouds $\mathcal{P}_0$ and $\mathcal{P}_1$, we wish to interpolate and render them to produce the image at a novel view and time t. Inspired by prior work [2, 21], we propose a depth-aware bidirectional splatting technique. In particular, we first obtain the 3D location of every point (in both point clouds) at time t by displacing it according to its associated scene flow scaled by t: $\mathbf{x}_{0\to t} = \mathbf{x}_0 + t\mathbf{u}_0$, $\mathbf{x}_{1\to t} = \mathbf{x}_1 + (1-t)\mathbf{u}_1$. The displaced points and their associated features from each direction ($0 \to t$ or $1 \to t$) are then separately splatted into the target viewpoint using differentiable point-based rendering [48], which results in a pair of rendered 2D feature maps $\mathbf{F}_{0\to t}, \mathbf{F}_{1\to t}$ and depth maps $\mathbf{D}_{0\to t}, \mathbf{D}_{1\to t}$. To combine the two feature maps and decode them to a final image, we linearly blend them based on spatial-temporal cues. Our general principles are: 1) if t is closer to 0 then $\mathbf{F}_{0\to t}$ should have a higher weight, and vice versa, and 2) for a 2D pixel, if its splatted depth $\mathbf{D}_{0\to t}$ from time 0 is smaller then the depth $\mathbf{D}_{1\to t}$ from time 1, $\mathbf{F}_{0\to t}$ should be favored more, and vice versa. Therefore, we compute a weight map to linearly blend the two feature and depth maps as follows:

$$\mathbf{W}_t = \frac{(1-t)\cdot \exp(-\beta \cdot \mathbf{D}_{0\to t})}{(1-t)\cdot \exp(-\beta \cdot \mathbf{D}_{0\to t}) + t \cdot \exp(-\beta \cdot \mathbf{D}_{1\to t})} \tag{1}$$

$$\mathbf{F}_t = \mathbf{W}_t \cdot \mathbf{F}_{0\to t} + (\mathbf{1} - \mathbf{W}_t) \cdot \mathbf{F}_{1\to t} \tag{2}$$

$$\mathbf{D}_t = \mathbf{W}_t \cdot \mathbf{D}_{0\to t} + (\mathbf{1} - \mathbf{W}_t) \cdot \mathbf{D}_{1\to t}. \tag{3}$$

Here $\beta \in \mathbb{R}_+$ is a learnable parameter that controls contributions based on relative depth. Finally, $\mathbf{F}_t$ and $\mathbf{D}_t$ are fed to a network that synthesizes the final color image.

3.5. Training

We train the feature extractor, image synthesis network, and the parameter β on two video datasets to optimize the rendering quality, as described below.

Training datasets. To train our system, we ideally would use image triplets with known camera parameters, where each triplet depicts a dynamic scene from a moving camera, so that we can use two images as input and the third one (at an intermediate time and novel viewpoint) as ground truth. However, such data is difficult to collect at scale, since it either requires capturing dynamic scenes with synchronized multi-view camera systems, or running SfM on dynamic videos shot from moving cameras. The former requires a time-consuming setup and is difficult to scale to in-the-wild scenarios, while the latter cannot guarantee the accuracy of estimated camera parameters due to moving objects and potentially insufficient motion parallax. Therefore, we found that existing datasets of this kind are not sufficiently large or diverse for use as training data. Instead, we propose two sources of more accessible data for joint training of motion interpolation and view synthesis.

The first source contains video clips with small camera motions (unknown pose). We assume that the cameras are static and all pixel displacements are induced by scene motion. This type of data allows us to learn motion interpolation without the need for camera calibration. The second source is video clips of static scenes with known camera motion. The camera motion of static scenes can be robustly estimated using SfM and such data gives us supervision for learning novel view synthesis. For the first source, we use Vimeo-90K [51], a widely used dataset for learning frame interpolation. For the second source, we use the MannequinChallenge dataset [14], which contains over 170K video frames of humans pretending to be statues captured from moving cameras, with corresponding camera poses estimated through SfM [56]. Since the scenes in this dataset including people are (nearly) stationary, the estimated camera parameters are sufficiently accurate for our purposes. We mix these two datasets to train our model.

Learnable components. Our system consists of several modules: (a) monocular depth estimator, (b) color and depth inpainter, (c) 2D feature extractor, (d) optical flow estimator and (e) image synthesis network. We could conceptually train this whole system, but in practice we train only modules (c), (d), and (e), and use pretrained state-of-the-art models [31, 38] for (a) and (b). This makes training less computationally expensive, and also avoids the need for the large-scale direct supervision required for learning high-quality depth estimation and RGBD inpainting networks.

Training losses. We train our system using image reconstruction losses. In particular, we minimize perceptual loss [9, 55] and l_1 loss between the predicted and ground-truth images to supervise our networks.

4. Experiments

4.1. Implementation details

For the feature extractor, we use ResNet34 [5] truncated after `layer3` followed by two additional up-sampling layers to extract feature maps for each RGB layer, which we augment with a binary mask to indicate which pixels are covered (observed or inpainted) in that layer. For the image synthesis network, we adopt a 2D U-Net architecture. For the optical flow estimator we use a pre-trained RAFT network [42] and fine-tune its weights during training. We use Pytorch3D [33] for differentiable point cloud rendering. Rather than using a fixed radius for all points, we set the radius of a point proportionally to its disparity when rendering a target viewpoint. This prevents foreground objects from

becoming semi-transparent due to gaps between samples when the camera zooms in.

We train our system using Adam [10], with base learning rates set to 10^{-4} for the feature extractor and image synthesis network, and 10^{-6} for the optical flow network [42]. We train our model on 8 NVIDIA V100 GPUs for 250k iterations for $\sim$ 3 days. We decrease the learning rates exponentially during the optimization. Each training batch contains 8 triplets randomly sampled from the Vimeo-90K [51] and MannequinChallenge datasets [14]. Within each triplet, the start and end images are used as input and the intermediate frame is used as ground truth. To train on MannequinChallenge, we must calibrate the monocular depth maps so that they align with the SfM point clouds. We estimate a disparity scale and shift for each depth map to minimize the MSE error between it and the depths of recovered SfM points. We discard sequences with large alignment errors during training. Please refer to the supplement for additional details.

4.2. Baselines

To our knowledge, there is no prior work that serves as a direct baseline for our new task of space-time view synthesis from the near-duplicate photos. One might consider dynamic-NeRF approaches [15,27,29,49] as baselines. However, these all require dense input views with known camera parameters and sufficient motion parallax, and thus do not apply to our scenario. Instead, as in NSFF [15], we found that we can combine individual methods to form baselines for our method. We describe three such baselines below.

Naive scene flow. As a simple baseline, we augment monocular depth with optical flow to get scene flow. Specifically, we first compute the monocular depths of the two views using DPT [31], and lift them into 3D to get two colored point clouds. We then use 2D optical flows generated by RAFT [42] to find pixels with mutual correspondences and compute their scene flows in the forward and backward directions. The two colored point clouds are then separately rendered to the target viewpoint at the intermediate time, producing two RGB images. Finally, we linearly blend the two rendered images based on the time t to obtain the final view. Note that this baseline does not perform inpainting.

Frame interpolation $\rightarrow$ 3D photo. Existing methods for frame interpolation and novel view synthesis can be combined to form a baseline for our task. Specifically, to synthesize an image at the novel time and viewpoint, we first adopt a state-of-the-art frame interpolation method, XVFI [39], to synthesize a frame at the intermediate time. We then apply 3D photo inpainting [38] to turn the interpolated frame into an inpainted LDI and render it from a desired viewpoint through a constructed mesh. For a fair comparison, we upgrade the 3D photo method to use the state-of-the-art monocular depth backbone DPT [31], i.e., the same monocular depth predictor we use in our approach.

3D photo $\rightarrow$ frame interpolation. This baseline reverses the order of operations in the aforementioned method. First, we apply the 3D photo [38] to each of the near-duplicates and render them to the target viewpoint separately. We then apply XVFI [39] to these two rendered images to obtain a final view at intermediate time t.

4.3. Comparisons on public benchmarks

Evaluation datasets. We evaluate our method and baselines on two public multi-view dynamic scene datasets: the NVIDIA Dynamic Scenes Dataset [52] and the UCSD Multi-View Video Dataset [16]. The NVIDIA dataset consists of 9 scenes involving more challenging human and non-human motions captured by 12 synchronized cameras at 60FPS. The UCSD dataset contains 96 multi-view videos of dynamic scenes, which capture diverse human interactions in outdoor environments. The videos are recorded by 10 synchronized action cameras at 120FPS. We run COLMAP [36] on each of the multi-view videos (masking out dynamic components using provided motion masks) to obtain camera parameters and sparse point clouds of the static scene contents.

Experimental setup. To evaluate rendering quality, we sample a triplet (two input and one target view) every 0.5 seconds from the multi-view videos. In each triplet, we select the two input views to be at the same camera viewpoint and two frames apart, and the target view to be the middle frame at a nearby camera viewpoint. We compare the prediction with the ground truth at the same time and viewpoint. All methods we evaluate use monocular depths that are only predicted up to an unknown disparity scale and shift. To properly render images into the target viewpoint and compare with the ground truth, we need to obtain aligned depth maps that are consistent with the reconstructed scenes. Similar to Sec. 4.1, we align the predicted depths with the depth from SfM point clouds. Please refer to the supplement for more detail.

Quantitative comparisons. We evaluate the rendering quality of each method using three standard error metrics: PSNR, SSIM and LPIPS [55]. Tab. 1 shows comparisons between our method and the baselines. Our method consistently outperforms the baselines in all error metrics. In particular, our LPIPS scores are significantly better, suggesting better perceptual quality and photorealism of rendered images for our approach. Note that all the methods have relatively low PSNR/SSIM because these metrics are sensitive to pixel misalignment, and inaccurate geometry from monocular depth networks can cause the rendered images to not fully align with the ground truth. But since all methods use DPT [31] depths, this issue does not affect the relative comparisons.

Qualitative comparisons. We show qualitative comparisons on the UCSD dataset in Fig. 4. Our method generates the fewest artifacts while preserving the most details in the scene. The naive scene flow baseline produces noticeable holes. Applying 3D Photos and then frame interpolation

Method	NVIDIA Dynamic Scene [52] PSNR↑	SSIM↑	LPIPS ↓	UCSD Multi-View Video [16] PSNR↑	SSIM↑	LPIPS↓
Naive Scene Flow	19.34	0.681	0.177	23.60	0.837	0.120
Frame Interpolation [39] → 3D Photo [38]	21.01	0.676	0.189	25.70	0.852	0.123
3D Photo [38] → Frame Interpolation [39]	21.18	0.681	0.192	25.96	0.858	0.126
Ours	**21.72**	**0.702**	**0.145**	**26.54**	**0.864**	**0.078**

Table 1. **Quantitative comparisons of novel view and time synthesis.** Our method outperforms all the baselines in all error metrics. See Sec. 4.2 for the descriptions of baselines.

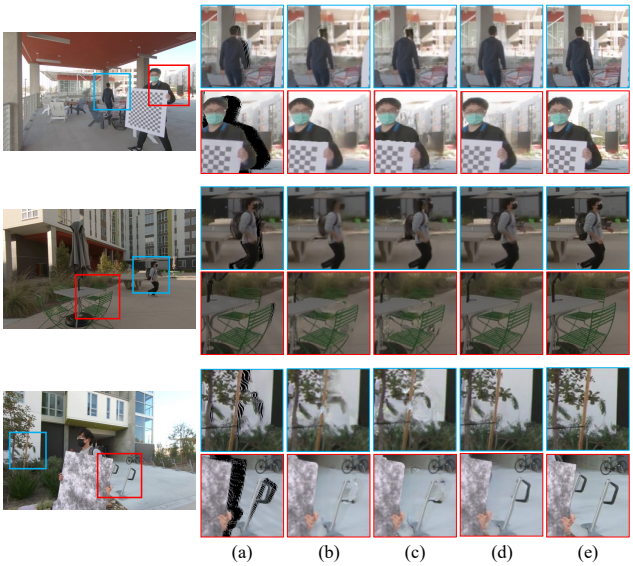


Figure 4. **Qualitative comparisons on the UCSD dataset [16].** From left to right are (**a**) naive scene flow, (**b**) frame interpolation [39] → 3D Photo [38], (**c**) 3D Photo [38]→ frame interpolation [39], (**d**) our method, and (**e**) ground truth.

leads to blurry disoccluded regions as the frame interpolator [39] is not trained to interpolate between inconsistently inpainted images. Applying frame interpolation and then 3D Photos leads to strong flickering artifacts due to inconsistent inpainting in each frame (see supplement video).

4.4. Comparisons on in-the-wild photos

We also evaluate our approach and the baselines qualitatively on in-the-wild near-duplicate photos. We collected these photos from our colleagues and their friends and families and obtained their consent to present these photos in this manuscript. We show comparisons of views generated by each method in Fig. 5. In particular, we show two different kinds of camera motions, zooming in and tracking, and rendering a novel view at intermediate time $t = 0.5$. Our method achieves overall better rendering quality with fewer visual artifacts, especially near moving objects and occlusion boundaries. We refer readers to the supplementary video for better visual comparisons of these generated 3D Moments.

	PSNR↑	SSIM↑	LPIPS↓
No features	21.16	0.693	0.173
No inpainting	21.33	0.685	**0.145**
No bidirectional	21.56	0.694	0.151
Full model Ours	**21.72**	**0.702**	**0.145**

Table 2. **Ablation studies on the NVIDIA dataset [52].** Each component of our system leads to an increase in rendering quality.

4.5. Ablations and analysis

Ablation studies. We conduct ablation studies to justify our design choices, as shown in Tab. 2. For "No features", instead of learning features we directly use RGB colors from the input photos to splat and render novel views. For "No inpainting", we train the system without inpainting color and depth in our LDIs and rely on the image synthesis network to fill in disoccluded regions in each rendered view separately (prone to temporal inconsistency). For "No bidirectional warping", we use only single-directional scene flow from time 0 to time 1.

Performance. Our method can be applied to new near-duplicate photo pairs without requiring test-time optimization. We test our runtime on an NVIDIA V100 GPU. Given a duplicate pair of images with resolution 768 × 576, it takes 4.48s to build LDIs, extract feature maps, and build the 3D feature scene flow. These operations are performed once for each duplicate pair. The projection-and-image-synthesis stage takes 0.71s to render each output frame.

5. Discussion and Conclusion

We presented a new task of creating 3D Moments from near-duplicate photos, allowing simultaneous view extrapolation and motion interpolation for a dynamic scene. We propose a new system for this task that models the scene as a pair of feature LDIs augmented with scene flows. By training on both posed and unposed video datasets, our method is able to produce photorealistic space-time videos from the near-duplicate pairs without substantial visual artifacts or temporal inconsistency. Experiments show that our approach

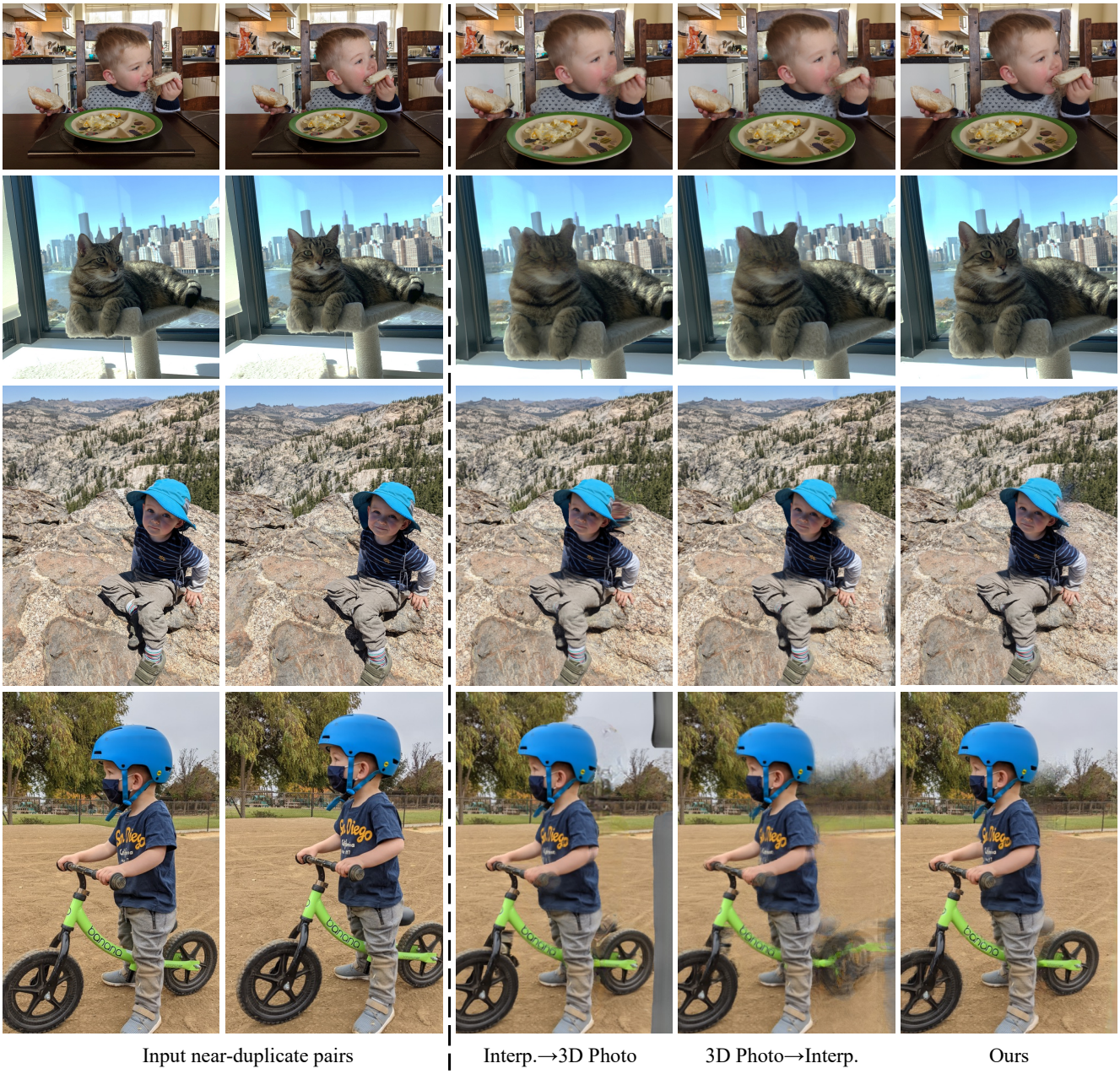


Figure 5. **Qualitative comparisons on in-the-wild photos.** Compared with the baselines, our approach produces more realistic views with significantly fewer visual artifacts, especially in moving or disoccluded regions. Please see the supplemental video for animated comparisons.

outperforms the baseline methods both quantitatively and qualitatively on the tasks of space-time view synthesis.

Limitations and future work. Our method inherits some limitations of monocular depth and optical flow methods. Our method does not work well for photos with complex scene geometry or semi-transparent objects. In addition, our method tends to fail in the presence of large and non-linear motions as well as challenging self-occlusions, such as hands. Please refer to the supplementary video for failure cases. Future work includes designing an automatic selection scheme for photo pairs suitable for 3D Moment creation, automatically detecting failures, better modeling of large or non-linear motions, and extending the current method to handle more than two near-duplicate photos.

Acknowledgements. We thank Richard Tucker, Tianfan Xue, Andrew Liu, Jamie Aspinall, Fitsum Reda and Forrester Cole for help, discussion and support.

References

[1] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *CVPR*, pages 5366–5375, 2020. 2

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *CVPR*, June 2019. 2, 5

[3] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: Implicit neural view-, light-and time-image interpolation. *ACM TOG*, 39(6), 2020. 2

[4] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM TOG*, 39(4), July 2020. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[6] Jun ho Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *ICCV*, 2021. 2

[7] V. Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Philemon Kaeser, William T. Freeman, D. Salesin, Brian Curless, and Ce Liu. SLIDE: Single image 3d photography with soft layering and depth-aware inpainting. In *ICCV*, 2021. 2, 4

[8] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*, pages 9000–9008, 2018. 2

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 5

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[11] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Péter Vajda, Ayush Saraf, and Michael F. Cohen. One shot 3d photography. *ACM Transactions on Graphics (TOG)*, 39:76:1 – 76:13, 2020. 1, 2, 3

[12] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *CVPR*, pages 5316–5325, 2020. 2

[13] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, S. Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *ArXiv*, abs/2103.02597, 2021. 2

[14] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *CVPR*, pages 4521–4530, 2019. 2, 5, 6

[15] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 1, 2, 6

[16] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3d mask volume for view synthesis of dynamic scenes. *ArXiv*, abs/2108.13408, 2021. 6, 7

[17] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[18] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation, 2020. 2

[19] Oded Maimon and Lior Rokach. *Data Mining And Knowledge Discovery Handbook*. 2005. 4

[20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 2

[21] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, pages 5436–5445, 2020. 2, 5

[22] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *CVPR*, pages 2270–2279, 2017. 2

[23] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, pages 261–270, 2017. 2

[24] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting adaptive convolutions for video frame interpolation. *arXiv preprint arXiv:2011.01280*, 2020. 2

[25] Simon Niklaus, Long Mai, Jimei Yang, and F. Liu. 3d ken burns effect from a single image. *ACM TOG*, 38:1 – 15, 2019. 1, 2

[26] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *ECCV*, pages 109–125. Springer, 2020. 2

[27] Keunhong Park, U. Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martín Brualla. Deformable neural radiance fields. In *ICCV*, 2021. 1, 2, 6

[28] Keunhong Park, U. Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *SIGGRAPH Asia*, abs/2106.13228, 2021. 1, 2

[29] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2, 6

[30] M. Usman Rafique, Hunter Blanton, Noah Snavely, and Nathan Jacobs. Generative appearance flow: A hybrid approach for outdoor view synthesis. In *BMVC*, 2020. 2

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 3, 4, 5, 6

[32] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 2020. 3

[33] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[34] Chris Rockwell, David F. Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021. 2

[35] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021. 2

[36] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 6

[37] Jonathan Shade, Steven Gortler, Li-wei He, and Rick Szeliski. Layered depth images. In *SIGGRAPH*, 1998. 3

[38] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, pages 8028–8038, 2020. 1, 2, 3, 4, 5, 6, 7

[39] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: extreme video frame interpolation. In *ICCV*, 2021. 2, 6, 7

[40] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, pages 175–184, 2019. 2

[41] Timo Stich, Christian Linz, Georgia Albuquerque, and Marcus Magnor. View and time interpolation in image space. In *Computer Graphics Forum*, volume 27, pages 1781–1787. Wiley Online Library, 2008. 2

[42] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020. 3, 5, 6

[43] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)*, 2020. 2

[44] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering, 2021. 2

[45] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, June 2020. 2

[46] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, pages 302–317, 2018. 2

[47] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[48] Olivia Wiles, Georgia Gkioxari, R. Szeliski, and J. Johnson. Synsin: End-to-end view synthesis from a single image. *CVPR*, pages 7465–7475, 2020. 2, 5

[49] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 1, 2, 6

[50] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation, 2019. 2

[51] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 127(8):1106–1125, 2019. 5, 6

[52] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, pages 5336–5345, 2020. 2, 6, 7

[53] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[54] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[55] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 5, 6

[56] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG*, 37:1 – 12, 2018. 2, 5

[57] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM TOG*, 23(3):600–608, 2004. 2