Implicit View-Time Interpolation of Stereo Videos using Multi-Plane Disparities and Non-Uniform Coordinates

Avinash Paliwal[1] Andrii Tsarov[2] Nima Khademi Kalantari[1]
[1]Texas A&M University [2]Leia Inc.
{avinashpaliwal,nimak}@tamu.edu, andrii.tsarov@leiainc.com

Abstract

In this paper, we propose an approach for view-time interpolation of stereo videos. Specifically, we build upon X-Fields that approximates an interpolatable mapping between the input coordinates and 2D RGB images using a convolutional decoder. Our main contribution is to analyze and identify the sources of the problems with using X-Fields in our application and propose novel techniques to overcome these challenges. Specifically, we observe that X-Fields struggles to implicitly interpolate the disparities for large baseline cameras. Therefore, we propose multi-plane disparities to reduce the spatial distance of the objects in the stereo views. Moreover, we propose non-uniform time coordinates to handle the non-linear and sudden motion spikes in videos. We additionally introduce several simple, but important, improvements over X-Fields. We demonstrate that our approach is able to produce better results than the state of the art, while running in near real-time rates and having low memory and storage costs.

1. Introduction

As the virtual reality (VR) and light field displays (e.g., Lume Pad [26]) become widespread, there is a growing need for capturing the appropriate content for these devices to provide an immersive virtual experience for the users. This necessitates capturing a scene from different views and at high frame rates. While this can be done using specialized hardware [3], such setups are usually bulky and expensive, and thus not suitable for an average user. To make the content capture widespread, we should focus on standard capturing devices like cellphone cameras. These devices, however, are typically equipped with only two cameras and are often not able to capture two videos at high frame rates. This necessitates interpolating across time and view to reconstruct high frame rate videos from dense views.

For a system to be practical and can be deployed on display devices with limited storage, memory, and computational capability, it should have a few properties: **1)** while a reasonable amount of post-processing can be done on a server, the approach should be able to generate results in real-time, **2)** it should not have a significant storage overhead on top of the input stereo video, and **3)** it should have a low memory cost.

Unfortunately, most existing approaches violate one or more criteria. For example, while the approaches based on multi-plane images (MPI) [47, 50, 55] can render novel views in real-time, they require storing the estimated MPIs for each frame (tens of megabytes per frame) and are additionally memory intensive. Moreover, to perform both view and time interpolations, these approaches need to be augmented with a video interpolation method which further adds to their memory and computational cost. The more recent approaches based on neural radiance fields (NeRF) [32] can perform both view and time interpolations [10, 29]. These methods encode the radiance field of a scene into a small network, and thus have a small storage and memory cost. However, they usually take a few seconds to render each novel view. Additionally, they require the cameras to be calibrated, and thus have difficulty handling general videos.

In this paper, we build upon X-Fields [2] that optimizes a coordinate-based network to learn an interpolatable implicit mapping between the input coordinates X (view or time) and the observed images. Once the optimization for a specific scene is performed, the network can be used to generate an image given any X coordinate. This approach satisfies all the properties, as the rendering is real-time (criterion 1) and the scene is encoded into a small network (criteria 2 and 3). Despite that, X-Fields struggles to produce reasonable results for the specific problem of view-time interpolation of stereo videos.

Our main contribution is to analyze X-Fields, identify the sources of the problems, and propose approaches that address these shortfalls. Specifically, we identify two major problems with X-Fields for stereo video interpolation. Our first observation is that X-Fields struggles to interpolate the disparities for cameras with large baselines. Second, we observe that linear motion in the input videos is critical for

X-Fields optimization to work, but non-linear motions are common in natural videos.

We address the first issue by proposing multi-plane disparities to reduce the spatial distance of the objects in the scene. Our approach makes it substantially easier for the network to interpolate the disparities, as the left and right disparities in different planes become closer to each other spatially. Moreover, we propose to address the second problem through a novel non-uniform time coordinate encoding method. We demonstrate that the implicit network is able to find a proper mapping between these non-uniform coordinates and the observations and has a significantly better interpolation capability.

Additionally, we propose a series of simple, but important, improvements such as additional regularization losses, learned blending, and positional encoding. Our approach has low memory and storage costs, and is able to reconstruct novel images in near real-time rates. Through extensive experiments we demonstrate that our method is significantly better than the state-of-the-art approaches. Code and supplementary materials are available on our project website at https://people.engr.tamu.edu/nimak/Papers/CVPR23StereoVideo.

2. Related Work

In this section, we briefly review the relevant view, time, and view-time synthesis methods.

2.1. View Synthesis

Novel view synthesis is a widely studied problem. A classical solution is to first reconstruct the 3D geometry of the scene, e.g., point clouds, and then render novel views based on the geometry [4, 5, 8, 15, 16, 25, 40, 45]. With the rise of deep learning, several approaches propose to handle this application using a neural network. For example, Flynn et al. [13] proposes a network based on plane sweep volumes, while Kalantari et al. [21] estimates the disparity and blending weights using two sequential networks. Zhou et al. [55] introduce multi-plane images (MPI), a flexible scene representation which is suitable for view synthesis. Several approaches [12, 28, 30, 31, 47, 50] propose various ways to extend this idea. The major problem with these approaches is significant storage and memory costs. Following the introduction of neural radiance field (NeRF) by Mildenhall et al. [32], a large number of approaches [9, 14, 17, 19, 37–39, 46] based on this idea have been proposed. While these approaches are powerful, they are typically slow to render novel images.

2.2. Time Interpolation

Most video interpolation approaches rely on optical flow to warp images and synthesize the interpolated frames. Recent state-of-the-art methods rely on deep learning for flow computation and image synthesis [1, 20, 34–36]. Several methods [7, 22] propose to directly generate the interpolated images without explicitly estimating a flow. Finally, a few methods [6, 43] adapt a network on the test example at hand by fine tuning it through the typical appearance loss.

2.3. View-Time Synthesis

Several recent approaches [10, 29, 41, 48, 51] extend NeRF to handle the additional time dimension and work on dynamic scenes. In addition to the common shortfalls of the NeRF-based approaches, these methods can only handle limited types of videos, as they require the cameras to be calibrated. Different from these approaches, Bemana et al. [2] propose X-Fields, a lightweight network capable of real-time view-time (as well as additional dimensions like light) interpolation. This approach essentially learns a per-scene mapping between the coordinates and 2D RGB images. We build upon X-Fields, but propose key ideas to significantly improve its performance on the problem of view-time interpolation of stereo videos.

3. Background

Given a set of images captured with different modalities (e.g., view and time), X-Fields [2] poses the problem as approximating the mapping between the input coordinates X and the corresponding images through a small coordinate-based neural network. To do so, X-Fields optimizes the network using the following objective:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \| f_\theta(\mathbf{y}_i) - f(\mathbf{y}_i) \|_1 \tag{1}$$

where $f(\mathbf{y}_i)$ is the observed image at coordinate $\mathbf{y}_i$ and $f_\theta(\mathbf{y}_i)$ is the approximated image using the network with weights θ. The approximated mapping function f_θ can then be used to reconstruct an image from any novel coordinate $\mathbf{x}$ (in the convex hull of the observed coordinates).

Instead of directly estimating the 2D RGB images using the network, X-Fields reconstructs the novel image by warping and combining the observed images in the neighborhood of the coordinate of interest. This is done by first estimating the Jacobian of the flows at each pixel. This Jacobian describes how a pixel in the coordinate of interest moves if, for example, the time coordinate changes. The Jacobian which is estimated by the network is defined as:

$$g_\theta(\mathbf{x})[\mathbf{p}] = J(\mathbf{x})[\mathbf{p}] = \frac{\partial \mathbf{p}(\mathbf{x})}{\partial \mathbf{x}}, \tag{2}$$

where g_θ is the network and $J(\mathbf{x})[\mathbf{p}]$ denotes the Jacobian at pixel $\mathbf{p}$ of coordinate $\mathbf{x}$. Using this Jacobian, the flow from pixel $\mathbf{p}(\mathbf{x})$ to the corresponding pixel $\mathbf{q}$ in coordinate $\mathbf{y}$ can be obtained by:

$$\mathbf{F}_\mathbf{y}(\mathbf{x})[\mathbf{p}] = \mathbf{p} + (\mathbf{y} - \mathbf{x}) J(\mathbf{x})[\mathbf{p}], \tag{3}$$

Figure 1. We show the images from the left and right views on the left. The interpolated middle images using only appearance loss (X-Fields) and our results are shown on the right. The stretching artifacts around the depth discontinuities because of the poorly estimated Jacobians can be observed in X-Fields results.

where $\mathbf{F}_{\mathbf{y}}(\mathbf{x})$ refers to the flow from the image at coordinate $\mathbf{x}$ to the one at $\mathbf{y}$. Using these flows, the observed neighboring images are warped to the coordinate of interest and are combined using a weight map, derived from the forward-backward flow consistency check.

4. Algorithm

Given a stereo video with N frames, our goal is to reconstruct images at novel views and times. We build upon X-Fields and attempt to learn an implicit mapping from the view-time coordinates $\mathbf{x}$ to the corresponding 2D RGB images $f(\mathbf{x})$. In our specific problem, the coordinates are 2 dimensional $\mathbf{x} = (u, t)$, where u is one dimensional view between and including the two views, and t denotes the one dimensional time coordinate.

We begin by discussing the view synthesis and time interpolation separately in Secs. 4.1 and 4.2, respectively. We then discuss the view-time interpolation in Sec. 4.3.

4.1. View Synthesis

Given a pair of stereo images at a specific frame, our goal here is to reconstruct novel views in between the two images. Since we would like our approach to have a low storage cost, we encode all the frames into a single neural network.[1] As the number of frames N can be large (90 for a 3 seconds video at 30 fps), we normalize the time coordinates to be between 0 and 1, i.e., $t = 0$ for the first frame and $t = 1$ for the last frame of the video. Moreover, we set the coordinates of the left and right views to $u_1 = -0.5$ and $u_2 = 0.5$, respectively.

Since our goal in this section is only view synthesis, our network needs to estimate a single channel Jacobian corresponding to the partial derivative of the horizontal displacement with respect to the view (i.e., disparity). With these settings we can optimize our network using Eq. 1 to perform this task. However, as shown in Fig. 1, X-Fields is not able to produce satisfactory results. Next, we discuss our approaches to significantly improve the results.

Jacobian Supervision: We observe that with the extremely sparse inputs in our application (two views), the appearance loss in Eq. 1 does not provide sufficient supervision to estimate reliable Jacobians. Note that in this case, only one image can be used to reconstruct the image $f_\theta(\mathbf{y}_i)$ at coordinate $\mathbf{y}_i$ during optimization. For example, to reconstruct the left image at frame $t = 0$, $(u, t) = (-0.5, 0)$, we can only use the right image at that frame $(0.5, 0)$. Unfortunately, minimizing the appearance loss forces the network to reconstruct the occluded regions in one view from the other (e.g., by texture stretching) by estimating Jacobians that do not properly represent the disparity (see Fig. 2). This negatively affect the quality of the interpolated results.

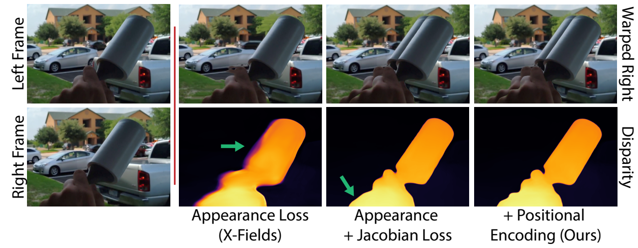

Figure 2. We show the images captured from left and right views of a particular frame on the left. Using only the appearance loss, the network tries to reconstruct the occluded regions by stretching the flow outside the depth discontinuities. With our additional Jacobian loss, we supervise the Jacobian in the occluded regions and produce better maps. Through positional encoding, the network is able to properly learn the complex boundaries.

To address this issue, we use an existing disparity estimation network (Li et al. [27] in our implementation) to constrain the estimated Jacobians in the occluded areas. Through training on a large number of scenes, this network is able to learn a prior and properly estimates the disparity in the occluded areas. Our key idea is to supervise our implicit network g_θ by only relying on the *guiding* disparity (from the pre-trained network) in the occluded areas, but use both the appearance and disparity supervisions in the other areas. To do this, we introduce the following objective:

$$\theta^* = \arg\min_\theta \sum_{j=1}^{2}\sum_{i=1}^{N} \|M^{\text{occ}} \odot (f_\theta(u_j, t_i) - f(u_j, t_i))\|_1 + \lambda\|g_\theta(u_j, t_i) - \tilde{J}(u_j, t_i)\|_1. \tag{4}$$

Here, $\tilde{J}(u_j, t_i)$ is the guiding Jacobian (disparity) and is obtained by passing the two views as the input to the pre-trained network. Specifically, to get $\tilde{J}(u_1, t_i)$ we pass the left and right images at frame t_i as the input and reverse the order of the images to obtain the other Jacobian $\tilde{J}(u_2, t_i)$. Moreover, M^{occ} is a binary mask with zero in the occluded areas and one in the other regions. We calculate this mask through forward-backward consistency check using the guiding Jacobians at the two views. This mask ensures that the warping loss (first term) is not used in the occluded areas. Finally, λ defines the weight of the second term, which is set to $20/w$ (where w is the frame width) in our implementation. In addition to improving the estimated

[1] This is in contrast to using a separate network for each stereo frame.

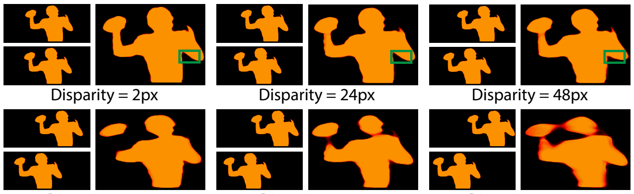

Figure 3. We show the network's interpolation ability on examples with varying disparities. For each case, we show the left and right Jacobians on the left and the interpolated Jacobian (for the middle view) on the right. As seen the quality of the interpolated Jacobians starts to degrade for cases with 48 pixels disparity and above. Note that the gap under the arm (shown in green box) becomes smaller in the case with 48 pixels disparity.

Jacobians in the occluded areas (see Fig. 2), the disparity supervision speeds up the convergence of the optimization and helps with the challenging cases like thin objects.

While the results produced with our method using the additional disparity supervision are reasonable, they sometimes contain ghosting and other artifacts around the depth discontinuities. We observe that these artifacts arise because the encoded Jacobians often lack sufficient details to represent the object boundaries (see Fig. 2). This is mainly because of the difficulty in encoding a large number of frames into a single small network. To address this issue, we apply positional encoding [32] (with 5 frequencies) to the time coordinates and use them instead as the input to our implicit network. Note that while positional encoding is proposed for multi-layer perceptron (MLP) networks, we observe that it improves the quality of estimated Jacobians using coordinate-based convolutional network as well, as shown in Fig. 2.

Multi-Plane Disparities: This approach is able to produce high-quality Jacobians at intermediate views for cases where the maximum disparity in the scene is small (e.g., small baseline cameras). However, the quality of the interpolated Jacobians for cases with large disparities (e.g., large baseline cameras) deteriorates, as shown in Fig. 3. This is because the objects with large disparities have a large spatial distance in the left and right Jacobians, making the implicit interpolation significantly more challenging.

We address this issue by reducing the spatial distance between the objects in the encoded left and right Jacobians. Specifically, we propose multi-plane disparities to represent the Jacobians at each coordinate. As shown in Fig. 4, instead of directly estimating the Jacobian, our network $g_\theta(u,t)$ estimates a set of Jacobians $J_{d_k}(u,v)$, each at a predefined disparity, $d_1, \cdots d_K$. The final Jacobian is then estimated from the multi-plane disparities as follows:

$$J(u,t) = r(g_\theta(u,t)) = \max_{d_1,\cdots d_K} s(J_{d_k}(u,t), d_k u), \quad (5)$$

where $s(J_{d_k}(u,t), d_k u)$ shifts the Jacobian at plane k, $J_{d_k}(u,t)$, $d_k u$ pixels to the left.

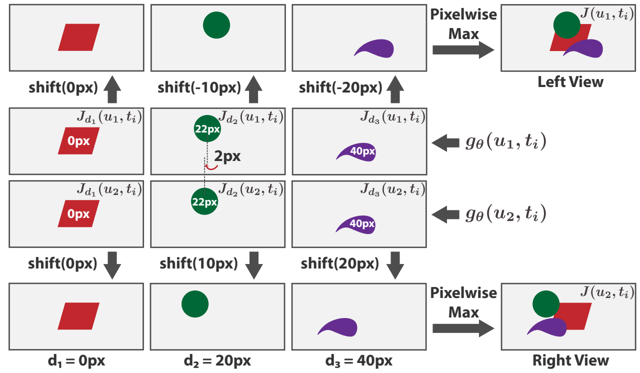

Figure 4. We demonstrate our multi-plane disparities with three planes at 0, 20, and 40 pixels. Our network estimates Jacobians on three planes, each encoding the objects with disparities in the proximity of their pre-defined disparity. For example, the object with 22 pixels disparity (green circle) is encoded into the plane $d_2 = 20$. We reconstruct the final disparity by shifting each plane $d_k u$ pixels to the left and computing the pixel wise max on the shifted planes. For example, to reconstruct the Jacobian at the left view ($u_1 = -0.5$), we shift the plane at $d_2 = 20$ equal to -10 pixels to the left. Because we move the objects to their correct location using the shift, the objects in the encoded multi-plane Jacobians at the left and right views are spatially close. This significantly improves the interpolation ability of our network.

Since we use $u_1 = -0.5$ and $u_2 = 0.5$ as the left and right view coordinates, through this process, the left and right Jacobians at each plane, $J_{d_k}(u_1,t)$ and $J_{d_k}(u_2,t)$, are shifted equal to $d_k/2$ pixels in the opposite directions. Therefore, as shown in Fig. 4, an object with d_k pixels disparity will be at the same spatial location in the encoded left and right Jacobians at plane k, since the object will be moved to the correct location using the shift operator. This significantly enhances the interpolation quality as the network, in this case, encodes left and right multi-plane disparities that contain objects at small spatial distances.

Moreover, since the objects that are closer to the camera have larger disparities, we obtain the final Jacobian by selecting the plane with maximum disparity at each pixel (see Eq. 5). This allows the network to reconstruct the unselected regions in a desired manner. The additional flexibility, provided by the max operator, makes it superior to other choices, such as summation (see comparisons in the supplementary materials).

Our network estimating the multi-plane disparities can be optimized using Eq. 4, with a small modification; instead of directly estimating the Jacobian using the network, we use the network to estimate the multi-plane disparities and reconstruct the Jacobian using Eq. 5. However, with such an optimization, there is no mechanism to enforce the network to utilize all the planes appropriately. For example, the network could potentially only use one of the planes to estimate the left and right Jacobians. In this case, our network will still not be able to properly interpolate the Jacobians, as shown in Fig. 5.

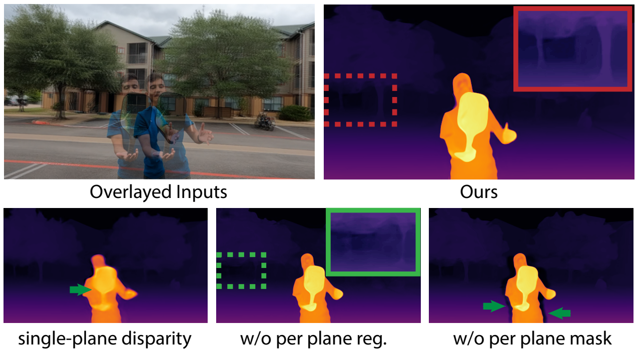

Figure 5. We show the impact of our multi-plane disparity, as well as the per plane regularization term and per plane mask in Eq. 6. Single-plane disparity fails to reconstruct the details during interpolation. Without the per plane regularization, the network does not utilize all the planes effectively. Therefore, it is not able to produce reasonable results in the regions corresponding to the unused planes (background in this case). Not using the per plane mask leads to halo artifacts around foreground objects. See supplementary Fig. 5 for more detailed intermediate results.

To address this issue, we propose to apply a regularization loss between the estimated and guiding Jacobians at each plane. As shown in Fig. 6, we compute the per plane guiding Jacobians by selecting disparities in the proximity of the plane's disparity. Specifically, to compute the guiding Jacobian at plane k (i.e., $\tilde{J}_{d_k}(u_j, t_i)$), we only select the disparities in the range $d_k - l/2$ and $d_k + l/2$, where l is the distance between consecutive disparities (e.g., 20 when the planes are at 0, 20, 40, etc.). Our final loss using this additional regularization is as follows:

$$\begin{aligned}\theta^* = \arg\min_{\theta} \sum_{j=1}^{2} \sum_{i=1}^{N} \| M^{\text{occ}} \odot (f_\theta(u_j, t_i) - f(u_j, t_i)) \|_1 \\ + \lambda \| r(g_\theta(u_j, t_i)) - \tilde{J}(u_j, t_i) \|_1 \\ + \gamma \sum_{k=1}^{K} \| M_k^{\text{disp}} \odot \left(s(J_{d_k}(u_j, t_i), d_k u_j) - \tilde{J}_{d_k}(u_j, t_i) \right) \|_1. \end{aligned} \tag{6}$$

where γ is the weight of the per plane regularization term and we set it to $1/w$ (where w is the frame width) in our implementation. Note that we use the shifted estimated Jacobians in the regularization term to align them with the per plane guiding Jacobian. Moreover, we use a per plane mask M_k^{disp} to ensure that this regularization is only applied in the regions where the guiding disparity has valid content. As shown in Fig. 6, we obtain the binary per plane mask by setting the regions with valid content to one and the remaining areas to zero. Without this mask, the network will be forced to match the guiding Jacobians, even in the areas without a valid content. This negatively affects the quality of the results, as shown in Fig. 5.

Blending: Using the optimized network, we can obtain a Jacobian at any coordinate. This Jacobian can then be used to obtain the flows to the left and right views using Eq. 3. The flows in turn can be used to warp the images to the novel coordinate. X-Fields [2] uses weight maps computed based on the forward-backward flow consistency to combine the images. However, their blending weights cannot fully avoid introducing the residual warping artifacts to the reconstructed image (see Fig. 7). This is particularly a problem in stereo video interpolation since only one image is used to reconstruct the other view. As such the blending weights are not utilized during optimization.

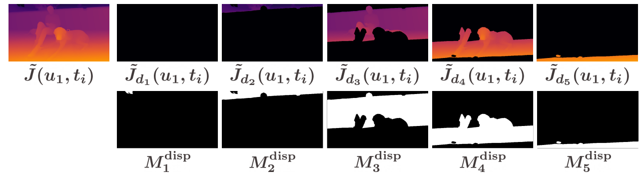

Figure 6. On the top, we show the guiding disparity (leftmost) along with the per plane guiding disparity images. On the bottom, we show the corresponding masks for each plane.

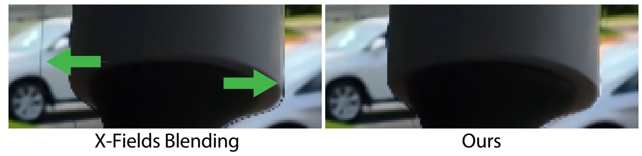

Figure 7. We show the interpolated images generated using the X-Fields blending weights and our learned weights. Our approach does not have the distracting artifacts around the boundaries.

To address this problem, we separately train a small network on a large dataset that takes the two inputs, warped images, as well as the corresponding flows and estimates a weight map. Using this map we obtain the reconstructed image at coordinate (u, t_i) as:

$$f_\theta(u, t_i) = \frac{(1-c) W_l I_l + c W_r I_r}{(1-c) W_l + c W_r}, \tag{7}$$

where $c = u - u_0$, while W_l is the estimated weight map for the left image and $W_r = 1 - W_l$. Moreover, I_l and I_r are the warped left and right views using the estimated Jacobian at coordinate (u, t_i).

4.2. Time Interpolation

The goal here is to reconstruct an image at a novel time coordinate from the two neighboring frames. Similar to the view synthesis case, we encode the entire stereo video into a single neural network to ensure low storage cost. Note that the network in this case estimates a two channel Jacobian at each coordinate, corresponding the partial derivative of the displacement in the x and y directions with respect to time. To handle this application, we apply all the enhancements from view synthesis (except multi-plane disparity, since it is specific to view synthesis), as we observe they improve the quality of the results. Specifically, we minimize the following loss, which is slightly different from the loss in Eq. 4:

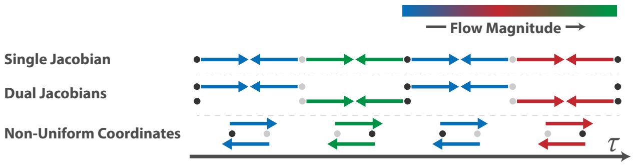

Figure 8. We show the frames with a black or a gray dot and the two flows to the next and previous frames with two color coded arrows (color representing their magnitude). Note that we are illustrating the ideal flows that need to be encoded into the neural network. Natural videos have non-linear motions, and thus it is difficult to represent the two flows at each frame using a single Jacobian. A straightforward way to address this problem is to estimate two different Jacobians at each frame (dual Jacobians). However, as discussed in Sec. 4.2, this approach (the same as single Jacobian) will have difficulty handling motion spikes (green). With our proposed non-uniform coordinates, we use two different coordinates to estimate the previous and next Jacobians at each frame. The large unused regions in-between allow the network to smoothly accommodate motion spikes.

$$\theta^* = \arg\min_\theta \sum_{j=1}^{2}\sum_{i=1}^{N} \| \left(f_\theta(u_j, t_i) - f(u_j, t_i)\right) \|_1 + \lambda \|g_\theta(u_j, t_i) - \tilde{J}(u_j, t_i)\|_1. \quad (8)$$

The main differences here are that we do not have an occlusion mask M^{occ} and our guidance Jacobian $\tilde{J}$ (estimated using Zhang et al.'s approach [53]) has two channels. This flow supervision significantly enhances the results as the exposure in different frames varies slightly which makes it difficult for the appearance loss (with brightness constancy assumption) to find appropriate Jacobians. Note that we do not use the occlusion mask, since for time interpolation, the image at the novel coordinate is generated by combining the warped previous and next frames. Therefore, we assume that all the areas are visible at least in one of the neighboring frames. We also apply positional encoding to the time coordinates (10 frequencies) and use the learned blending weights as opposed to the weights computed by forward-backward flow consistency [2].

Unfortunately, as shown in Fig. 9 ("Single Jacobian"), even with the additional enhancements, our system is not be able to produce satisfactory results in some cases. Specifically, we observe that the system struggles in cases where the motion becomes non-linear. This is because we use three consecutive frames during optimization; specifically, we minimize the error between $f(u_j, t_i)$ and the reconstructed frame using the previous $f(u_j, t_{i-1})$ and next $f(u_j, t_{i+1})$ frames. The main assumption here is that the motion is linear, i.e., a single two channel Jacobian at u_j, t_i can describe the flow to the previous and next frames. However, this assumption is typically violated in natural videos.

A straightforward way to handle this problem is to estimate two sets of Jacobians at each coordinate (see Fig. 8), $J_b(u_j, t_i)$ and $J_a(u_j, t_i)$, where they denote the Jacobians to the previous and next frames , respectively. The two Jacobians can be estimated using a single network (with a four channel output) or two separate networks. Although this approach can handle the cases with non-linear motions reasonably well (see "Dual Jacobians" in Fig. 9 - top), in some cases it produces results with severe ghosting (Fig. 9 - bottom). To understand the reason, we show the average flow magnitude of the guidance flow (obtained with a pretrained network) for the consecutive frames of one view in Fig. 10. As seen, natural videos, in particular the ones captured with handheld devices, often contain motion spikes. Unfortunately, both the single and dual Jacobians solutions have difficulty producing Jacobians that can properly estimate such spikes. This is because the network smoothly interpolates between the observations, and thus the spikes will be over-smoothed.

Figure 9. We show two interpolated frames (top and bottom) generated using single Jacobian, dual jacobians, and our approach with non-uniform coordinates. On the top, we show that because of non-linearity of motion in the scene, with a single Jacobian we generate results with severe ghosting. This problem can be resolved with dual Jacobians. On the bottom, we show a frame with motion spikes. Both single and dual Jacobians fail to properly estimate the large motion in this case. Our approach with non-uniform coordinates, however, produces high-quality results in both cases.

We address this issue by proposing to encode the Jacobians using non-uniform time coordinates τ (see Fig. 8). Specifically, we estimate all the Jacobians to the next frame (J_a) at the original time coordinate ($\tau = t$ for J_a). However, for the Jacobian to the previous frame (J_b) we shift the coordinate closer to the previous frame, i.e., $\tau = t - \alpha$ for J_b, where α is set to 0.9 in our implementation. With this transformation, we are able to effectively encode the Jacobians (see Fig. 10) and produce high-quality interpolated frames (Fig. 9). In this case, the large unused regions in-between allow the network to smoothly ramp up and down to and from the motion spikes.

4.3. View-Time Synthesis

Our goal here is to combine the view and time interpolation systems to be able to synthesize an image at any novel view-time coordinates given a stereo video. While we can perform both view and time synthesis by encoding the view and time Jacobians into one network, we observe that the

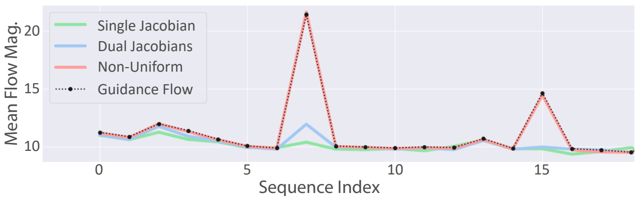

Figure 10. We show the mean flow magnitude (obtained using a pre-trained flow network) for the consecutive frames in a video. As seen, natural videos often contain motion spikes, specially for handheld cameras. Single (X-Fields) and dual Jacobians are unable to properly estimate these spikes. Our method with non-uniform coordinates produces results that closely match the guidance flow.

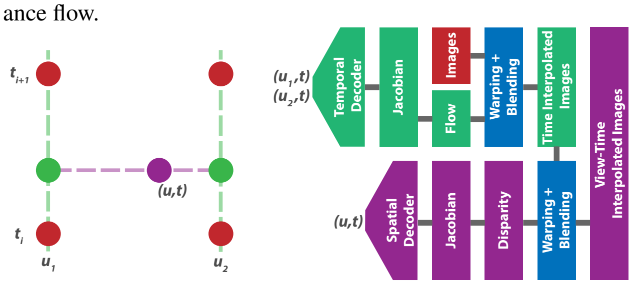

Figure 11. Overview of our view-time interpolation system.

two types of Jacobians often conflict with each other. This, unfortunately, negatively affects the quality of the results.

Therefore, we propose to perform the view-time synthesis in two stages (see Fig. 11). Given 4 frames neighboring the coordinate of interest (u, t), we first obtain the time interpolated images at coordinates (u_1, t) and (u_2, t). In the second stage, we perform view interpolation between these two images to calculate the final image at coordinate (u, t).

5. Implementation

We set the coordinates of the left and right views to $u_1 = -0.5$ and $u_2 = 0.5$, respectively. However, when passing these view coordinates to the network we scale them down by a factor of 30 (i.e., $u_1 = -1/60$ and $u_2 = 1/60$), as we empirically observe that with closer coordinates, the network is able to better interpolate between the Jacobians. This is because with closer coordinates the network tends to produce intermediate Jacobians that are correlated with the ones at the left and right views. However, after a certain point, the interpolation quality deteriorates as we bring the coordinates closer [11].

We use the convolutional decoder network as proposed by Bemana et al. [2] with a capacity factor of 16 for both view and time interpolations. Moreover, we use a UNet with 5 downsampling/upsampling layers to estimate the blending weights. We train the network for 10k iterations using the Adam optimizer [23] with a learning rate of 10^{-4} on the Vimeo90K [52] video dataset. As the loss, we use the L1 distance between the blended and ground truth images.

In our implementation, we further enhance the view weight maps (W_l and W_r in Eq. 7) by computing gradient of the flow in the x direction. We then threshold the derivative to identify the edge regions. We expand these edges horizontally by applying dilation-erosion (morphological operations) with an anisotropic kernel to create a residual mask. Finally, we combine this residual mask with the weight map through binary operations.

We perform our per-scene optimization for 100k iterations using Adam. We start with a learning rate of 10^{-4} and decrease it by a factor of 0.4 every 33k iterations to speed up convergence.

6. Results

We compare our approach against the approaches by Bemana et al. [2] (X-Fields) and Du et al. [10] (NeRF+T, NeRFlow). We also compare against a combination of MPI (for view interpolation) by Zhou et al. [55] and FILM by Reda et al. [42] (for time interpolation). We call this combined method FILM+MPI throughout this section. For X-Fields, we use a larger network than ours to ensure the network capacity is not a limiting factor for their performance. Du et al. first train a NeRF with an additional input for the time coordinate (NeRF+T) and then fine-tune it for dynamic scenes using pre-trained flows and depth maps (NeRFlow). We compare our method against both versions of this approach. For all the approaches, we use the source code provided by the authors. Here, we show results on a few scenes, but more comparisons, and the videos, are provided in the supplementary materials.

6.1. Qualitative Results

We capture a set of stereo videos with a variety of motions using a stereo GoPro camera rig. We show the results using this camera setup in the paper, but we also test our approach using Lume Pad [26] and provide the results in the supplementary video.

We show comparisons against several state-of-the art approaches on a few scenes in Fig. 12. For all the scenes, we show view-time interpolation at the middle of four observed view-time frames. As seen, other approaches produce results with noticeable ghosting and other artifacts, while our results are sharp and have clear boundaries.

6.2. Quantitative Results

We numerically compare our approach against the other methods on two lightfield video datasets, Sintel [24] and LFVID [44], in terms of PSNR, SSIM [49], and LPIPS [54]. The comparisons against FILM+MPI and X-Fields are shown in Table 1. As seen, our approach produces significantly better results than both of these methods across all the metrics. Note that, we excluded NeRF+T and NeRFlow from these comparisons as the camera calibration fails for some of the sequences (some scenes are captured with a tripod mounted camera).

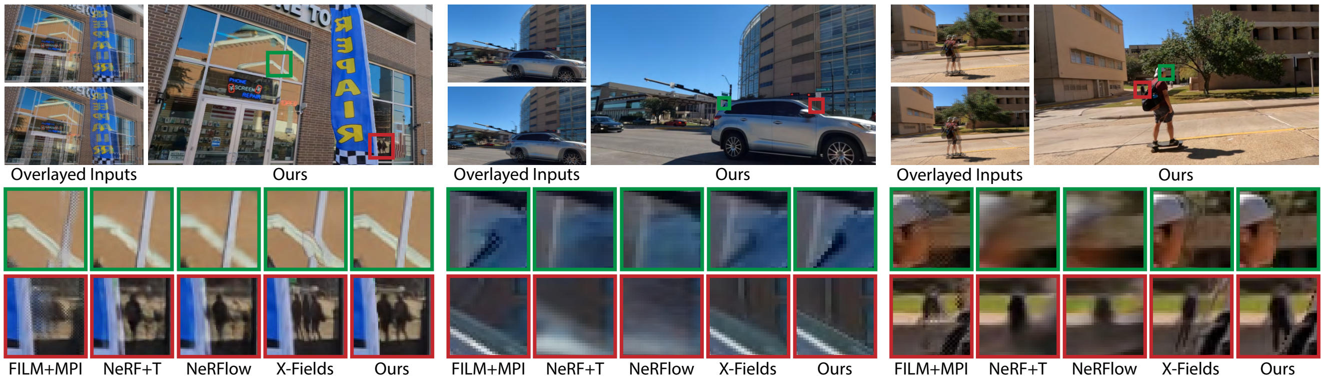

Figure 12. Comparison against several state-of-the-art methods on view-time interpolation. On the left we show the overlayed left and right views for two consecutive frames neighboring the coordinate of interest.

Table 1. View-time synthesis results on Sintel [24] and LFVID [44] datasets.

	Sintel			LFVID		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
FILM+MPI	24.39	0.782	0.1423	18.23	0.496	0.3046
X-Fields	25.38	0.814	0.1263	19.33	0.552	0.2365
Ours	**26.33**	**0.860**	**0.0989**	**23.49**	**0.696**	**0.1345**

6.3. Training and Inference Times

We compare the training (optimization) and inference speed of our approach against the other methods. The training times are obtained on a machine with an NVIDIA V100 GPU, while the inference speed is measured on a machine with a 2080Ti GPU. We provide the timings for a stereo video with 100 360p frames. Our approach takes roughly 120 minutes to train our view and time interpolation networks. In comparison, X-Fields requires 170 minutes of training, while NeRF+T and NeRFlow take around 5 and 25 hours, respectively. At run-time, our approach takes 44 ms to generate a novel view-time image. In contrast, X-Fields takes 26 ms, while NeRF+T and NeRFlow require 17 seconds to generate a single image. Note that, X-Fields is faster as it performs view-time synthesis using a single network, while ours requires evaluating the Jacobians using two separate networks. However, we believe given the improvement in quality, this is justified. Nevertheless, the performance can be significantly improved by optimizing the network architecture and implementing the approach on efficient frameworks like *tiny-cuda-nn* [33].

Note that, MPI does not perform any optimization and can render novel images in real-time given the MPI for each frame. However, the MPI for each frame is around 120 MB (12 GB for 100 frames), and thus their method is significantly storage and memory intensive. In comparison, our network takes around 12 MB of storage space for the entire 100 frames Additionally, MPI only performs view synthesis and we augmented this method with FILM, which takes roughly 0.5s per frame, to be able to handle view-time interpolation. While there are other approaches, such as RIFE [18], that can perform time interpolation at real-time, they often produce results with lower quality.

6.4. Limitations

Our approach uses Jacobian supervision, and thus the performance of our system depends on the quality of the guidance Jacobians (estimated with Li et al. [27] and Zhang et al. [53]). As such, poor quality guidance Jacobians can negatively affect our optimization. However, as discussed, pure per-scene optimization for such an ill-posed problem is not effective and incorporating the results of networks, trained over a large number of scenes is necessary. Additionally, our method will not be able to properly reconstruct regions that are occluded in both neighboring frames, e.g., left and right views. Such information, however, might exist in other frames in the video. Therefore, addressing this problem by combining the NeRF-based approaches with our method would be an interesting future research.

7. Conclusion

We present an approach to generate images from any novel view-time coordinates from an input stereo video. We analyze and identify the problems with using X-Fields in our application. We make two key observations based on our analysis: 1) the network struggles to interpolate the Jacobians for cases with large disparities and 2) the main assumption of X-Fields is linear motion which is violated in natural videos. Based on these observations, we propose multi-plane disparities and non-uniform time coordinates to improve the results. We demonstrate that our method significantly outperforms the state of the art.

8. Acknowledgement

The authors would like to thank the reviewers for their comments and suggestions. This work was funded by Leia Inc. (contract #415290).

References

[1] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[2] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6), 2020. 1, 2, 5, 6, 7

[3] 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 Transactions on Graphics (Proc. SIGGRAPH)*, 39(4):86:1–86:15, 2020. 1

[4] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 425–432, 2001. 2

[5] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3), 2013. 2

[6] Myungsub Choi, Janghoon Choi, Sungyong Baik, Tae Hyun Kim, and Kyoung Mu Lee. Scene-adaptive video frame interpolation via meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[7] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *AAAI*, 2020. 2

[8] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 11–20, 1996. 2

[9] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 2

[10] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 1, 2, 7

[11] Pedro Figueirêdo, Avinash Paliwal, and Nima Khademi Kalantari. Frame interpolation for dynamic scenes with implicit flow encoding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 218–228, January 2023. 7

[12] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[13] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deep stereo: Learning to predict new views from the world's imagery. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5524, 2016. 2

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

[15] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *ACM Trans. Graph.*, 36(6), 2017. 2

[16] Peter Hedman and Johannes Kopf. Instant 3d photography. *ACM Trans. Graph.*, 37(4), 2018. 2

[17] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5875–5884, October 2021. 2

[18] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 8

[19] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, October 2021. 2

[20] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[21] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 35(6), 2016. 2

[22] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv preprint arXiv:2012.08512*, 2020. 2

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 7

[24] Takahiro Kinoshita and Satoshi Ono. Depth estimation from 4d light field videos. In *International Workshop on Advanced Imaging Technology (IWAIT) 2021*, volume 11766. International Society for Optics and Photonics, 2021. 7, 8

[25] Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. Sampling based scene-space video processing. *ACM Trans. Graph.*, 34(4), 2015. 2

[26] Leia. Leia inc. https://www.leiainc.com/, 2022. 1, 7

[27] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16263–16272, 2022. 3, 8

[28] Qinbo Li and Nima Khademi Kalantari. Synthesizing light field from a single image with variable mpi and two network fusion. *ACM Transactions on Graphics*, 39(6), 12 2020. 2

[29] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2

[30] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3d mask volume for view synthesis of dynamic scenes. In *ICCV*, 2021. 2

[31] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[32] 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. In *ECCV*, 2020. 1, 2, 4

[33] Thomas Müller. Tiny CUDA neural network framework, 2021. https://github.com/nvlabs/tiny-cuda-nn. 8

[34] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1710, 2018. 2

[35] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[36] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *European Conference on Computer Vision*, pages 109–125, 2020. 2

[37] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 2

[38] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021. 2

[39] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9054–9063, June 2021. 2

[40] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6), 2017. 2

[41] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, June 2021. 2

[42] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision (ECCV)*, 2022. 7

[43] Fitsum A Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised video interpolation using cycle consistency. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[44] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. Dataset and pipeline for multi-view light-field video. In *CVPR Workshops*, 2017. 7, 8

[45] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006. 2

[46] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7495–7504, June 2021. 2

[47] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[48] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12959–12970, October 2021. 2

[49] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 7

[50] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8534–8543, June 2021. 1, 2

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

[52] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented

flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019. 7

[53] Feihu Zhang, Oliver J. Woodford, Victor Adrian Prisacariu, and Philip H.S. Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10807–10817, 2021. 6, 8

[54] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 7

[55] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4), 2018. 1, 2, 7