Abstract

In this paper, we propose an algorithm to interpolate between a pair of images of a dynamic scene. While in the past years significant progress in frame interpolation has been made, current approaches are not able to handle images with brightness and illumination changes, which are common even when the images are captured shortly apart. We propose to address this problem by taking advantage of the existing optical flow methods that are highly robust to the variations in the illumination. Specifically, using the bidirectional flows estimated using an existing pre-trained flow network, we predict the flows from an intermediate frame to the two input images. To do this, we propose to encode the bidirectional flows into a coordinate-based network, powered by a hypernetwork, to obtain a continuous representation of the flow across time. Once we obtain the estimated flows, we use them within an existing blending network to obtain the final intermediate frame. Through extensive experiments, we demonstrate that our approach is able to produce significantly better results than state-of-the-art frame interpolation algorithms.

1. Introduction

With the widespread availability of smartphones featuring increasingly powerful cameras, taking professional high resolution photos has become a simple press of a button. Often, people take many pictures in search for the best representation of a moment in terms of the expression, pose, lighting, and exposure. Interpolating these photos creates a video with an exciting effect that provides an appealing way of remembering key moments.

Existing video interpolation approaches [20, 42] often struggle to handle these cases because of significant camera and scene motion. Recently, Reda et al. [47] propose to overcome this challenge by leveraging a multi-scale feature extraction and flow estimation strategy. Specifically, they use a series of convolutional layers to construct a feature pyramid and use it to estimate a set of two flows from the in-between frame to the two input images at multiple scales. These flows are then used to warp the two images along with their features at each scale to the frame of interest. The warped features and images are then aggregated and combined to produce the final frame.

While this approach produces interpolated results with high quality, it is limited to cases where the two images have consistent brightness and illumination. However, in practice, the two images could have different brightness and illumination even if taken shortly apart, as shown in Fig. 1. Unfortunately, the approach by Reda et al. [47] is not able to handle these cases, producing results with unnatural motion and severe artifacts.
Our key observation is that the source of the problem is that their estimated flows degrade quickly in presence of illumination changes (see Fig. 1). This is largely because their network is trained on video datasets [60, 52] where the frames are captured in quick succession, and thus have consistent brightness. Therefore, the images with lighting variations fall outside the distribution of their training data. On the other hand, their blending, even though trained on video datasets, is usually able to generate pleasing intermediate images. The key to generating high-quality images thus lies in improving the quality of the estimated flows.

In this paper, we propose to address this problem by utilizing a pre-trained optical flow network. Existing optical flow methods [58, 61] are highly robust to even significant lighting changes, and thus are suitable for our application. The major challenge here is that these methods only estimate the flow between two images, but we need the flows from an intermediate frame to the two input images.

To overcome this challenge, we propose a per-scene optimization method (no training on large datasets) by utilizing implicit neural networks [35, 54]. Our key idea is that by encoding the bidirectional flows between the two input images into a coordinate-based network, we essentially obtain a continuous representation of flows across time. Therefore, we can use such a network to estimate the flows at any in-between time coordinate. To be able to properly estimate the intermediate flows, we use a hypernetwork that takes the time coordinate and estimates the weights of a coordinate-based neural network. The coordinate-based network then estimates the flow at each pixel by taking the pixel coordinate as the input. We optimize the hypernetwork using the bidirectional flows and then estimate any in-between flows by passing the appropriate time coordinate to this optimized hypernetwork. We then use these intermediate flows with Reda et al.’s blending network to generate the final intermediate images.

We show that our method outperforms existing approaches on a wide range of challenging scenes with large lighting variations and motions (see Figs. 1 and 4 and the supplementary video). Furthermore, we justify our design choices through extensive experiments.

2. Related Work

In this section, we review the frame interpolation methods, as well as the approaches in image morphing, a relevant but different problem. We also briefly discuss implicit neural representations as we utilize them in our work.

2.1. Frame Interpolation

In recent years, deep learning methods have become popular due to their effectiveness in handling challenging scenarios like scenes with large complex motions. Niklaus and Liu [37] use a pretrained flow network to warp the existing frames and then use a context-aware blending network to synthesize the interpolated frame. Similarly, Jiang et al. [25] generate interpolated frames at arbitrary time by estimating the flows to the intermediate frame. Wenbo et al. [3] utilize a depth estimation network to handle the occlusions. Niklaus and Liu [38] use forward warping with a synthesis network to generate interpolated frames.

Moreover, Park et al. [41] propose a model based on bilateral motion estimation to generate high quality warped frames for blending. They further enhance this approach [42] by computing asymmetric bilateral fields to account for non-linearities in the scene. Huang et al. [20] directly estimate the intermediate flows by using a privileged distillation scheme during training. Reda et al. [47] propose a unified network consisting feature pyramid and flow extraction with fusion components to handle scenes with large motions. This method along with a few other recent approaches [29, 19] focus on improving the quality on high-resolution videos. Furthermore, a couple of approaches [32, 48] propose to improve the performance of the supervised methods by further training the system in an unsupervised manner.

In contrast to these methods, several approaches propose to directly estimate the final image without explicitly estimating the required flows. For example, Niklaus et al. [39, 40] use adaptive convolution kernels to generate the intermediate frame from the neighboring images. Choi et al. [11] use PixelShuffle [51] with channel attention [59] to directly synthesize the middle frame. Gui et al. [16] blend deep features and Kalluri et al. [26] utilize 3D space-time convolutions for interpolation. Unfortunately, all of these techniques, as well as the flow-based methods, train their system on sequences with consistent illumination, and thus are not suitable for our application.

Related to our work, Bemana et al. [5] learn an implicit mapping between view-time-light coordinates and input images by optimizing a convolutional network. However, they use a loss between the input and warped images for optimization, and thus their main assumption is brightness constancy which is invalid in our cases. We address this problem by utilizing a pre-trained flow network that is highly robust to illumination changes.

2.2. Morphing

Similar to our application, image morphing methods produce a series of images to smoothly transition between two input images. Most algorithms [4, 8, 30, 49, 31] achieve this by first computing a set of sparse correspondences between the two images, and then using them to warp the images to an intermediate frame. These warped images are then combined to create a morphed image. These methods are typically best suited for images of different scenes. Since
they typically utilize sparse correspondences, their transitions for our examples (images of the same dynamic scene) are not sufficiently detailed to produce appealing effects.

Several approaches [50, 12] propose to handle this application using patch-based optimization systems. Similarly, these methods are suitable for different objects/scenes and are not able to produce visually pleasing result for images of the same object/scene. Recent breakthroughs in deep learning and generative adversarial networks [6, 27] have enabled efficient and high quality morphing by interpolating in the latent space [1, 21, 44]. However, these approaches mostly work for a single or few objects (e.g., faces, cats, cars) and are not general.

2.3. Implicit Neural Representations

A large number of recent methods have used neural networks as a memory-efficient continuous function approximator for implicit representation of images [54, 57, 36, 15, 9] and videos [7, 62, 54]. 3D objects via signed distance functions [24, 46, 43, 2, 53, 54] or occupancy networks [33, 10], and radiance fields [35, 14, 22, 18, 56]. Novel input encodings [35, 57] and activation functions [54] have been instrumental in enabling the encoding of high frequency details in compact networks for these applications. We build upon these advances, but use implicit neural representations for optical flow interpolation.

3. Method

Given a pair of images of a dynamic scene, $I_{t_0}$ and $I_{t_1}$, captured under different conditions, e.g., different exposures, the goal of our method is to reconstruct an image at time $t$ between the two images, where $t \in [t_0, t_1]$. Most existing frame interpolation methods, and in particular the state-of-the-art method of Reda et al. [47], break down this process into flow estimation and blending components. Specifically, they first compute a set of flows from the intermediate frame at time $t$ to the two input images, $F_{t \rightarrow t_0}$ and $F_{t \rightarrow t_1}$. They then use these flows to backward warp the images/features to the intermediate frame and combine them to reconstruct the final image $I_t$.

Unfortunately, these approaches are not able to properly handle cases with illumination variation mainly because the quality of the estimated flows degrades quickly in absence of brightness constancy. This is expected as these methods train their system on video datasets that contain minimal lighting variations between the neighboring frames.

To address this problem, we propose to utilize the powerful optical flow estimation methods [58, 61] that are highly robust to these illumination changes. The major challenge is that using these optical flow methods, we can only estimate the bidirectional flows between the inputs, $F_{t_0 \rightarrow t_1}$ and $F_{t_1 \rightarrow t_0}$, but we require estimating the intermediate flows.

We propose to address this challenge by implicitly interpolating the bidirectional optical flows using a coordinate-based neural network. Once the intermediate flows are estimated, we incorporate them in the blending network by Reda et al. [47] to estimate the final image. The overview of our system is shown in Fig. 2. Next, we discuss our implicit flow interpolation and blending approaches.

Discussion: One might attempt to train existing video interpolation methods on datasets with lighting variations to handle this application. However, constructing a dataset of input images with realistic illumination changes and their corresponding intermediate ground truth image is difficult. Moreover, even if the dataset can be reconstructed, it might be challenging to design a network that can outperform state-of-the-art optical flow methods. Finally, by using existing flow estimation methods, we have the ability to replace them with newer and better methods to further improve our results in the future. We also note that we considered forward warping, instead of backward warping, using Niklaus et al.’s method [38] to avoid the need for computing the intermediate flows. However, their approach requires computing an importance mask to properly handle the overlapping regions. Unfortunately, this importance mask is computed with the assumption of brightness constancy which is invalid in our cases.

3.1. Implicit Flow Interpolation

Our goal here is to estimate the intermediate flows, $F_{t \rightarrow t_0}$ and $F_{t \rightarrow t_1}$, from the bidirectional flows between the two input images, $F_{t_0 \rightarrow t_1}$ and $F_{t_1 \rightarrow t_0}$, generated by an existing pre-trained flow network. In our system, we use the optical flow method of Teed et al. [58] because of its ability to generate high quality flows in challenging cases. As discussed, we propose to estimate the intermediate flows implicitly through a coordinate-based neural network.

A coordinate-based network finds a mapping between an input coordinate and the corresponding output at that coordinate, i.e., $y = f_\theta(x)$, where $\theta$ refers to the weights of the network. This network can then be optimized on a set of input output pairs $(x_i, y_i)$ by minimizing a loss to find optimal network weights $\theta$. Through this optimization, the data will be encoded into the weights of the neural network. The key idea is that, once this optimization is performed, we can evaluate the network at any in-between coordinate to obtain the corresponding output. The network will essentially interpolate the output at the observed coordinates to generate the intermediate results.

In our application, the input coordinates are 3-dimensional, $x = (x, y, t)$, where $x$ and $y$ are the spatial and $t$ is the time coordinate. The output, on the other hand, is a 2D flow (in horizontal and vertical directions) at each coordinate $y = F(x, y, t)$. Note that we use flows that are normalized by the difference in the time coordinates as the out-
put to our network, i.e., \( F(x, y, t_0) = F_{t_0 \to t_1}(x, y)/(t_1 - t_0) \) and \( F(x, y, t_1) = F_{t_1 \to t_0}(x, y)/(t_0 - t_1) \). This is because the original flows are in opposite directions \((t_0 \text{ to } t_1)\) and \((t_1 \text{ to } t_0)\) and cannot be interpolated. By normalizing the flows with their coordinate difference the direction of the two flows become consistent. We encode these normalized flows into our coordinate-based network using the following objective:

\[
\theta^* = \arg \min_{\theta} \left\| \sum_{i=1}^w \sum_{j=1}^h \sum_{k=0}^1 f_{\theta}(x_i, y_j, t_k) - F(x_i, y_j, t_k) \right\|_2,
\]

where \(w\) and \(h\) are the width and height of the images (and similarly the flows), respectively.

Once this optimization is performed, we can evaluate the network at any arbitrary in-between time coordinate \(t \in [t_0, t_1]\) to estimate the intermediate flow. Note that the network produces a normalized flow at each coordinate. We then convert this to the two intermediate flows as follows:

\[
\begin{align*}
F_{t \to t_0}(x, y) &= (t - t_0) \times f_{\theta}(x, y, t) \\
F_{t \to t_1}(x, y) &= (t - t_1) \times f_{\theta}(x, y, t).
\end{align*}
\]

For our network, we use SIREN, as proposed by Sitzmann et al. [55], with 5 hidden layers each containing 128 neurons. Moreover, we set the frequency of the sinusoidal activation functions to 10. As shown in Fig. 3, this approach (Single SIREN) is not able to properly interpolate the two input flows. This is because a coordinate-based network, generates the in-between results by “averaging” the data at the observed nearby coordinates. Therefore, the network reconstructs the intermediate flow by combining the two flows \((t_0 \text{ and } t_1)\) at the same spatial coordinates. Essentially, the network generates the “average” of the two encoded flows resulting in blurry intermediate flows. We address this problem by encoding the time coordinate using a hypernetwork, as discussed next.

### 3.2. Time Encoding Through A Hypernetwork

To properly interpolate the intermediate flow, we need to “average” the shape of the two input flows. Our main ob-
servation is that by encoding data into a coordinate-based network, the shape will be essentially represented using the weights of the network. Therefore, we can encode the normalized bidirectional flows into two separate coordinate-based networks. The networks in this case, only takes the spatial coordinates as the input \( f_0(x_i,y_j) \) and the two networks are independently optimized to encode the flows at \( t_0 \) and \( t_1 \). The optimized weights, \( \theta_0 \) and \( \theta_1 \), can then be linearly interpolated to obtain the representation of the intermediate flow \( \theta_i \). The interpolated weights can be used to generate the in-between flow. However, as shown in Fig. 3, this strategy (Two SIRENs) does not produce the desired interpolation as the two flows are independently encoded and the two representations, \( \theta_0 \) and \( \theta_1 \), are not interpolatable.

To address this problem, we propose to estimate the weights of the coordinate-based network using a hypernetwork [17] that takes the time coordinate as the input, i.e., \( \theta = f_\phi(t) \). We encode both normalized flows into our system through the following objective:

\[
\phi^* = \arg \min_{\phi} \sum_{i=1}^{w} \sum_{j=1}^{h} \sum_{k=0}^{1} || f_\phi(x_i,y_j) - F(x_i,y_j,t_k) \||_2,
\]

where \( \theta = f_\phi(t_k) \).

In this case, we encode the two flows simultaneously and since the representations (weights \( \theta_0 = f_\phi(t_0) \) and \( \theta_1 = f_\phi(t_1) \)) are estimated using a single hypernetwork, they are closer together in the high dimensional space, and thus are interpolatable. The main difference with respect to Two SIRENs strategy is that, here, the estimated representations (weights) are produced by the same network (hypernetwork). While, in theory, \( \phi^* \) that produces the same \( \theta_0 \) and \( \theta_1 \) as in the two SIREN strategy is a valid minimizer of Eq. 3, by using a small hypernetwork and initializing the weights to small values, we usually converge to a solution that produces highly correlated SIREN weights in practice.

Once the hypernetwork is optimized, we generate the final intermediate flows by evaluating the hypernetwork at the in-between time coordinate \( \theta_{\frac{k}{2}} = f_\phi(t) \) and using the calculated weights in Eq. 2. We also experimented with first estimating the weights at \( t_0 \) and \( t_1 \) using our hypernetwork (i.e., \( \theta_0 = f_\phi(t_0) \) and \( \theta_1 = f_\phi(t_1) \)) and then linearly interpolating them to obtain \( \theta_{\frac{k}{2}} \), but the results were similar.

Our hypernetwork is composed of a set of fully-connected networks with one hidden layer of size 128 with ReLU activation that maps the time coordinate to the weights in each layer of our coordinate-based network (SIREN). Moreover, we use \( t_0 = 0 \) and \( t_1 = 0.1 \) to further force the network to produce highly correlated weights (see the effect of coordinate distance in Fig. 7).

In summary, using the small hypernetwork along with the close time coordinates, we confine the space of possible weights, which is essential for high-quality interpolation. As shown in Fig. 3, our system with a hypernetwork is able to properly encode the two flows and reconstruct an intermediate flow.

### 3.3. Blending

As discussed, we use our interpolated flows with the pretrained blending network by Reda et al. [47] (FILM) to generate the final interpolated images. While FILM’s blending system is trained on standard datasets with mostly constant illumination, we observe that it produces visually pleasing interpolation between images with varying lighting. To incorporate our estimated intermediate flows into their system, we first generate a flow pyramid by downsampling our estimated intermediate flows to multiple scales. We use bilinear interpolation to downsample the flows and divide the magnitude of the flows by the scale factor. Once we obtain the pyramid of the two flows, we use them to warp the feature pyramid (calculated with FILM’s feature extractor), as well as the input images, to the intermediate frame. Finally, we pass all the warped features and images to FILM’s fusion network to obtain the final result.

### 4. Results

We implement our model in PyTorch [45] and utilize Torchmeta [13] for our hypernetwork. We leverage pretrained flow estimation network of Teed et al. [58] (RAFT) and the blending network by Reda et al. [47] (FILM). Our solution uses the sintel checkpoint for RAFT and the style checkpoint for FILM’s blending. Specifically, RAFT’s sintel checkpoint has been trained on pairs of images with their corresponding ground truth flow from the Sintel dataset [23]. RAFT applies various data augmentations to ensure robustness to a variety of distortions. We optimize our model using Adam [28], with the default parameters \( \beta_1 = 0.9 \) and \( \beta_2 = 0.999 \). We train for 10K iterations on a single A100 GPU using a learning rate of \( 1e^{-6} \).

### 4.1. Comparisons

We compare our algorithm to state-of-the-art video frame interpolation approaches by Park et al. [42] (ABME) and Reda et al. [47] (FILM). We use the source code provided by the authors for both approaches.

**Quantitative:** Numerical evaluation of the quality of the interpolated images is challenging as there are no datasets containing input images with lighting variations and their corresponding ground truth intermediate images. While we could potentially use existing datasets and apply various perturbations (e.g., hue) to the input images, constructing the corresponding ground truth intermediate image remains a challenge as these perturbations are non-linear.

Therefore, we only numerically evaluate the quality of the intermediate flows. To do so, we use the two video frame interpolation datasets of Xiph 2K and 4K [38], as
Table 1: We numerically compare our interpolated flows against the ones by Reda et al. [47] in terms of average end-point-error (EPE). For these comparisons we use Xiph 2K and 4K [38], as well as Sintel [23].

<table>
<thead>
<tr>
<th></th>
<th>Xiph 2K</th>
<th>Xiph 4K</th>
<th>Sintel</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILM</td>
<td>13.97</td>
<td>37.34</td>
<td>12.9</td>
</tr>
<tr>
<td>Ours</td>
<td>3.4</td>
<td>17.46</td>
<td>5.16</td>
</tr>
</tbody>
</table>

well as the Sintel dataset [23]. For each input image pair, we apply various photometric augmentations by randomly perturbing brightness, contrast, saturation, and hue. We use Pytorch’s COLORJITTER with brightness 0.4, contrast 0.4, saturation 0.4, and hue 0.5/π. We then use the perturbed images as the input to estimate the intermediate flows which are then compared to the reference flows. The reference intermediate flows for the Sintel dataset are provided, but for Xiph 2K and 4K we simply use the RAFT flows between the intermediate and two input clean (unperturbed) images as the reference. For Xiph 2k and 4k, we skip six frames when creating the image pairs (e.g. 1-7, 2-8, etc.) to increase the amount of motion of the scenes, while we skip one frame for the Sintel dataset.

Table 1 shows the comparison against the approach by Reda et al. [47] in terms of average end-point error (EPE). Note that we do not include the method by Park et al. [42] since their approach does not explicitly estimate flows. As seen, our interpolated flows are significantly better than the estimated intermediate flows by Reda et al. We show visual comparisons on a few images from all datasets in the supplementary material.

**Qualitative:** We perform qualitative comparisons on several challenging scenes captured using a smartphone at resolution 2016 × 1512. To show the robustness of our approach, we capture both indoor and outdoor scenes with image pairs taken at different times of the day or seconds apart. We compare the estimated middle image by all the approaches on 6 scenes in Fig. 4, but encourage the readers to see our supplementary video.

We begin by examining the BABY scene which contains non-rigid motion in an indoor setup. The baby transitions from a shaded curious pose to a partially lit smiley expression. This scene illustrates how natural lighting can change significantly even for photos taken seconds apart. ABME produces blurry results in the moving regions. While FILM generates a sharper interpolation, it deforms the baby’s head. Our method preserves the baby’s appearance, producing a realistic intermediate image in terms of expression, motion, and lighting.

The HOUSE scene demonstrates our method’s ability of interpolating images under extremely different lighting conditions. The two images are captured from a mostly static scene, but at different times of the day (morning and evening). ABME generates a blurry interpolation with ghosting artifacts, while FILM produces an unnatural interpolation by introducing dark patches throughout the image. Our method, on the other hand, is able to interpolate intermediate images with reasonable quality because of the ability of our system to interpolate high quality flows.

The HUG scene contains both moving subjects and significant camera motions. The slight lighting variations on the trees and shadows combined with the large motions make this scene extremely challenging for the other approaches. In contrast, RAFT is able to estimate high quality flows and our approach properly interpolates the intermediate flows to produce results without objectionable artifacts. Similarly, the LAMP scene, while static, contains significant camera motion and has been captured with different exposures (see the sky in the input images) making it a challenging scene for the other methods. Although our approach produces slight ghosting artifacts around the object boundaries, our results are still plausible and significantly better than the other techniques.

The TREE scene contains significant subject motions and lighting variations (see the building roof and shadows in the input). ABME blurs out the entire frame, while FILM severely distorts the background. In contrast, our approach produces a high-quality interpolation by smoothly warping the subject while maintaining a coherent background. Finally, although the LADY is a relatively easy scene, ABME produces a blurry background and FILM is not able to properly reconstruct the gaps in the chair. Our approach, however, produces a high quality results without any objectionable artifacts.

### 4.2. Ablation Experiments

**Effect of Hypernetwork:** We begin by evaluating the effect of the hypernetwork on the quality of the interpolated flows. In Fig. 3, we show its effect on a synthetic example. The effect on a real example is shown in Fig. 5. As seen, while all the approaches are able to encode the input flows at times t₀ and t₁ with similar quality, only our method with a hypernetwork is able to properly reconstruct the interpolated flow.

**Effect of ω:** Next, we evaluate the effect of the frequency ω of the sinusoidal activation functions of SIREN in Fig. 6. The frequency is a key factor in properly encoding the data into a coordinate-based network. Higher frequencies are more suitable for signals containing a lot of details, while lower frequencies are more appropriate for smooth signals. As shown, frequencies between 8 to 12 produce reasonable results in our cases. We use ω = 10 in our implementation.

**Effect of Time Coordinates:** In Fig. 7, we show the effect of changing the time coordinates (t₀ and t₁), which are used as inputs to our hypernetwork. As seen, with a large distance between the two coordinates (0.2 and 0.5), our system is not able to produce high-quality intermediate flows.
Figure 4: We show comparisons against the state-of-the-art methods by Park et al. [42] (ABME) and Reda et al. [48] (FILM).
Figure 5: We evaluate the impact of the hypernetwork on the quality of the interpolated flow on a real example. All the approaches properly encode the flows at $t_0$ and $t_1$, but only our approach with the hypernetwork can produce an interpolated flow with high quality.

Figure 6: Effect of varying $\omega$ of SIREN on the interpolated flow at $t_{0.5}$.

This is because, in this case, the hypernetwork will not provide sufficient constraints and the estimated coordinate-based network weights for the two coordinates ($\theta_0$ and $\theta_1$) become independent. On the other hand, when the coordinates are too close to each other (distance of 0.02), the hypernetwork becomes overly restrictive and cannot estimate proper weights. Distances of 0.1 and 0.05 are ideal and produce the best quality.

Figure 7: Effect of changing the distance between the input time coordinates $t_0$ and $t_1$ on the quality of the interpolated flows. In all the cases, $t_0$ is equal to 0.

Figure 8: Comparison between the blending approaches by Huang et al. [20] (RIFE) and Reda et al. [48] (FILM) when used with our estimated flows.

4.3. Limitations and Future Work

Our method uses the flows estimated by RAFT [58], and thus the quality of our results depends on the accuracy of these predicted flows. While RAFT produces high quality flows in a large number of cases, it could potentially fail on challenging scenarios. In these cases, the flow artifacts may appear in our final results. However, since our approach allows us to use any optical flow method, as better flow estimation approaches are developed in the future, we can simply use them to further improve our results.

We also explored the idea of directly interpolating the images, instead of the flows, but were not successful based on our initial experiments. We believe this is because images are significantly more detailed than the optical flows. We leave a thorough investigation of this idea to the future.

5. Conclusion

We presented an approach to interpolate between a pair of images of a dynamic scene with lighting variations. We propose to do so by utilizing existing optical flow methods. To calculate the flows between an intermediate frame and the two input images, we interpolate the bidirectional flows estimated using a pre-trained flow network in an implicit manner. Specifically, we encode the bidirectional flows into a coordinate-based network and estimate the flows at any time, by passing the appropriate coordinate as the input. We use the estimated flows within an existing blending network to produce the final interpolated image. We show that our method is able to produce significantly better results than the state of the art on a wide range of challenging scenes.
References


[54] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representa-


