Multi-scale Intermediate Flow Estimation for Video Frame Interpolation

Zehua Fan[1], Feng Zhu[1,2], Lei Li[2] and Xiaoyang Tan[1]
[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Email: {fzh169,smc,x.tan}@nuaa.edu.cn
[2]Nanjing Research Institute of Electronics Engineering, Nanjing, China
Email: stone1743@126.com

***Abstract*—Video frame interpolation is one of the most challenging tasks in video processing, which aims to synthesize intermediate frames between consecutive frames. In this work, we propose a flow-based approach called Multi-scale Intermediate Flow Estimation (MIFE) to balance the fineness and estimation range of the flows. MIFE consists of two main modules. Specifically, (1) Refined Flow Estimation uses a shifted window to estimate low-resolution intermediate flows at three levels. The refined full-resolution flow of each level is a weighted combination of nearby low-resolution flows, where the weights are determined by the similarity scores of the input frames and the reliability scores of the flows. (2) Multi-scale Flow Fusion generates fusion masks based on the estimable flow range and the estimated flow size. It fuses three levels of flows and refines the results. Experimental results show that the proposed method achieves good performance on various datasets. The source code is available at https://github.com/fzh169/MIFE.**

***Index Terms*—video frame interpolation, video processing, flow estimation**

I. Introduction

Video frame interpolation aims to synthesize an intermediate frame between two consecutive video frames. It is a challenging problem in computer vision. This technology is widely used in frame rate up-conversion [1], video compression [2] and slow motion generation [3], [4]. In addition, some methods combining video frame interpolation with deblurring [5] or super-resolution [6] have achieved good results.

The popular video frame interpolation methods include phase-based [7], [8], kernel-based [9]–[12] and flow-based [13]–[17]. The phase-based methods predict the phase and amplitude values of the frames step by step, and then reconstruct the final image at different levels based on these predictions. They can accommodate large inter-frame motions but suffer from missing textures. The kernel-based methods adaptively estimate the large-size kernel for each pixel of the intermediate frames, and then perform a convolution operation on the kernels with the input frames to synthesize the intermediate frames. The motions that such methods can handle are limited by the size of the kernels. However, the mismatch between the size of the kernels and the motions leads to difficulties in practical terms.

Flow-based methods are the classical methods for video frame interpolation [18]–[20]. They usually consist of two steps: flow estimation and frame interpolation. For such methods, the quality of the interpolated frames depends heavily on the flow quality. To obtain more accurate flows, some methods [13], [16], [17] use pre-trained state-of-the-art flow estimation models as sub-structures. In recent years, flow-based methods have actively developed as the performance of flow estimation models [21]–[24] continues to improve. However, the pre-trained model cannot directly estimate the flows between the unavailable intermediate and input frames. Although some methods [3], [4], [15] try to approximate the intermediate flows using the flows between the input frames, the approximation error degrades the performance of the intermediate frames. In addition, labeling flows between consecutive frames is more difficult than directly acquiring multiple frames in a video.

In this work, our goal is to simultaneously cope with different sizes of motion at a low computational cost. We propose the Multi-scale Intermediate Flow Estimation (MIFE) to achieve this. MIFE consists of two main modules: Refined Flow Estimation (RFE) and Multi-scale Flow Fusion (MFF). In the RFE module, we first extract the three levels of features and estimate the flows using the correlation volume between the input frames. We introduce the shifted window proposed by Swin Transformer [25] to reduce the computational complexity. Then, we extract the similarity scores of the input frames using a convolutional network called ScoreNet. A weighted combination of the 3×3 low-resolution flows near each position yields the refined full-resolution flows. In the MFF module, we calculate the range of flows that can be estimated for each position in each level. Then, we generate a fusion mask based on the estimable flow range and the estimated flow size. Multi-scale flows are fused to obtain more accurate flows. We employ an extra network to refine the fused flows. Finally, we predict the candidate intermediate frames by backward warping and synthesize preliminary results using occlusion masks. To reduce the artifacts in the intermediate frames, we generate residual frames using warped pyramidal contextual features. The final result is the sum of the residual and preliminary intermediate frames.

Our model does not rely on other pre-training components and additional flow supervision. Experiments show that MIFE performs well on several public benchmarks compared to representative state-of-the-art frame interpolation methods. MIFE balances the fineness and estimation range of the flows to synthesize sharper intermediate frames under different motions.

II. Related Work

A. Video Frame Interpolation

Flow-based video frame interpolation methods have become very popular in recent years. The flow-based methods estimate the optical flows between the input frames and then generate an intermediate frame by forward warping or backward warping. Niklaus et al. [17] estimate the bidirectional optical flows between input frames using PWC-Net [23] and extract the context maps. Finally, they use a synthetic network to obtain the intermediate frame. However, multiple pixels in the input frames may map to the same target pixel in the intermediate frame during forward warping, resulting in holes and interpolation artifacts in overlapping regions [26]. Niklaus et al. [16] propose Softmax Splatting to combine overlapping pixels adaptively and thus improve the quality of the interpolated frame.

Other methods generate interpolated frames based on backward warping. For example, Jiang et al. [3] estimate the forward and backward flows separately. Then, they use a linear combination of two bidirectional flows as an initial approximation of the intermediate flow and improve it using U-Net. Reda et al. [27] propose synthesizing intermediate frames using unsupervised cyclic consistency. Backward warping does not create the problem of holes and overlapping pixels. However, it introduces approximation errors that degrade the performance of video frame interpolation because it needs to estimate the optical flows from the intermediate to the input frames, whereas the intermediate frame is unavailable.

On the other hand, kernel-based approaches have recently achieved significant advances. Niklaus et al. [9] first propose AdaConv, which treats frame interpolation as a local convolution of two input frames. They learn a spatially adaptive convolution kernel for each pixel using a CNN. AdaConv obtains high-quality results, but it is computationally too expensive. Therefore, Niklaus et al. [10] propose SepConv to solve this problem by predicting separable kernels. Inspired by the former, Lee et al. [11] further extend the intermediate flows to achieve adaptive collaboration of flows using deformable convolution kernels. Cheng et al. [12] extend the kernel-based method using deformable separable convolution and further propose EDSC [28] to generate interpolated frames at arbitrary times.

B. Swin Transformer

In order to reduce the computational complexity, we use the shifted window approach in the flow estimation. We are inspired by Swin Transformer, a generic backbone network recently proposed by Liu et al. [25]. It starts with small patches and gradually merges neighboring patches in deeper layers to build hierarchical representations. It improves efficiency by limiting the self-attentive computation to non-overlapping local windows and allowing cross-window connections through a shifted window scheme. This hierarchy has the flexibility to model at different scales. With these hierarchical feature mappings, Swin Transformer can easily utilize advanced techniques for intensive prediction. Since it only computes the self-attention within each local window and the number of patches in each window is fixed, its complexity is linear with the image size.

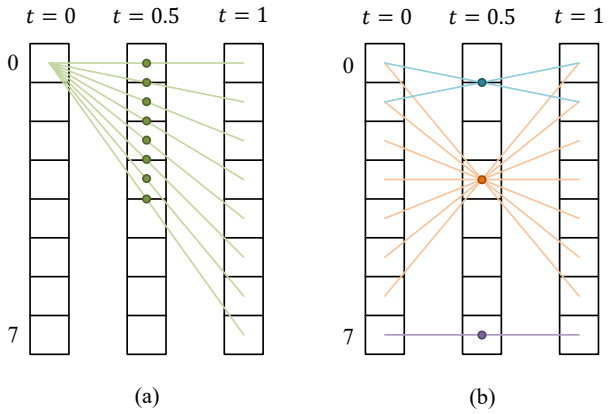

Fig. 1. Illustrations of the correlation pairs between consecutive frames in the one-dimensional case. (a) Example of correlation pairs. (b) All correlation pairs cross certain points.

III. Proposed Approach

A. Overview

Given a pair of consecutive frames I_0 and I_1, the goal of video frame interpolation is to synthesize frame I_t, where $t \in (0,1)$ is an arbitrary time position. Usually, t is set to 0.5, i.e., the target is the intermediate frame between I_0 and I_1. We show an overview of the proposed MIFE in Fig. 2. We first estimate three levels of intermediate flows $F^l_{0.5\to0}$, $F^l_{0.5\to1}$ and flows between input frames $F^l_{0\to1}$, $F^l_{1\to0}$, where $l \in \{1,2,3\}$. Then, we fuse the multi-scale flows to get more accurate flows. We estimate the occlusion mask $M \in [0,1]$ using the fused $F_{0\to1}$ and $F_{1\to0}$. We predict the preliminary intermediate frame $\widetilde{I}_{0.5}$ by the following equation:

$$\widehat{I}_{0.5\leftarrow0} = \overleftarrow{\omega}(I_0, F_{0.5\to0}) \tag{1}$$

$$\widehat{I}_{0.5\leftarrow1} = \overleftarrow{\omega}(I_1, F_{0.5\to1}) \tag{2}$$

$$\widetilde{I}_{0.5} = M \odot \widehat{I}_{0.5\leftarrow0} + (1-M) \odot \widehat{I}_{0.5\leftarrow1} \tag{3}$$

where $\overleftarrow{\omega}(\cdot,\cdot)$ is the backward warping and $\odot$ is the element-by-element multiplication. The backward warping can be implemented using bilinear interpolation and is differentiable. Finally, we introduce the warped contextual features $T_{0.5\leftarrow0}$ and $T_{0.5\leftarrow1}$ to reduce the artifacts of the synthesized intermediate frames.

B. Refined Flow Estimation

Feature Extractor. We apply Feature Extractor to I_0 and I_1 respectively, to obtain hierarchical features. Feature Extractor outputs three levels of features with resolutions of $1/2, 1/4$, and $1/8$ of the input frames. Each level of it includes one downsampling block and two residual blocks. We use layernorm to normalize the output features at each level.

Flow Estimation. We use flow $F^l_{0\to1}(x,y)$ to denote the difference in position of the point (x,y) in $t = 0$ to its

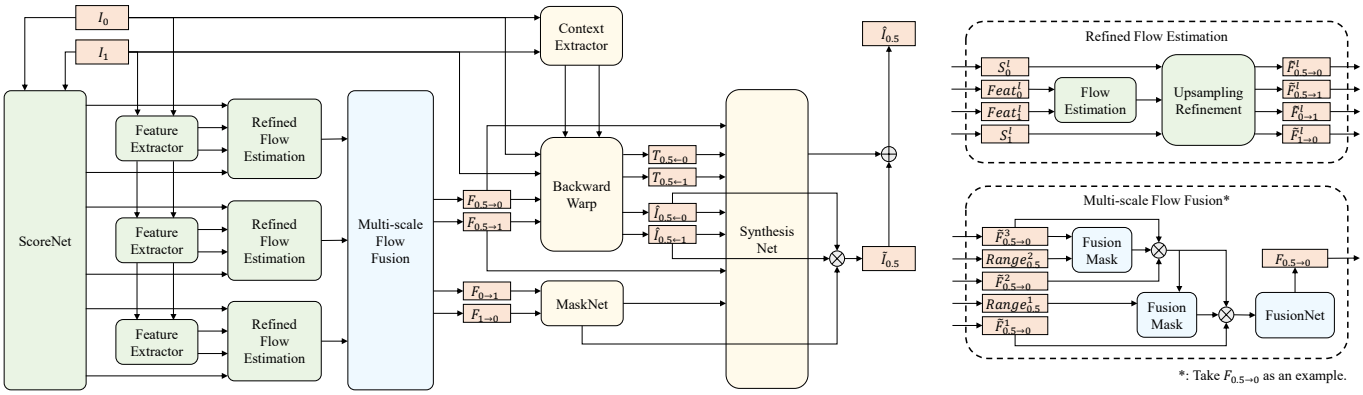

Fig. 2. The overall architecture of our network. Given a pair of consecutive frames I_0 and I_1, we first extract their hierarchical features and similarity scores. Using them, we estimate three levels of intermediate flows and flows between input frames. Then, we fuse multi-scale flows to obtain more accurate flows. We employ the fused flows to estimate occlusion masks and predict intermediate frames. Finally, we introduce warped contextual features to generate residual frames and add them to the preliminary intermediate frames to obtain the final result.

corresponding point in $t = 1$ at level l. If (x_0, y_0) is a point in $t = 0$ such that its corresponding point in $t = 1$ is (x_1, y_1), then:

$$F^l_{0\to1}(x_0, y_0) = (x_0 - x_1, y_0 - y_1) \tag{4}$$

Since the intermediate frames are unavailable, the intermediate flows $F^l_{0.5\to0}$ and $F^l_{0.5\to1}$ cannot be computed directly. Assuming that the motion between consecutive frames is linear, we approximate the intermediate flow using the following equation:

$$F^l_{0.5\to1}\left(\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2}\right) = (x_0 - x_1, y_0 - y_1) * 0.5 \tag{5}$$

The (x_1, y_1) corresponding to (x_0, y_0) is unknown, hence we approximate it by the point in $t = 1$ that has the maximum correlation with (x_0, y_0). Given the feature pairs $Feat^l_0$ and $Feat^l_1$, the correlation volume $C^l_{0,1}$ can be obtained by computing the dot product between all the feature pairs.

For the sake of description, we discuss the one-dimensional case, where only x exists. Fig. 1(a) illustrates the correlation pairs of point 0 in $t = 0$ with all points in $t = 1$. The green dots mark the positions corresponding to these correlation pairs in $t = 0.5$. We assume that $m = \frac{x_0+x_1}{2}$ is the point in $t = 0.5$. For a fixed m:

$$\begin{cases} F^l_{0.5\to1}(m) = (x^\star_0 - x^\star_1) * 0.5 \\ R^l_{0.5,1}(m) = C^l_{0,1}(x^\star_0, x^\star_1) \\ (x^\star_0, x^\star_1) = \underset{(x_0,x_1)\in\mathbb{P}}{\arg\max}\, C^l_{0,1}(x_0, x_1) \end{cases} \tag{6}$$

where $\mathbb{P} = \{(x_0, x_1)|x_0 \in t = 0, x_1 \in t = 1, (x_0, x_1)$ crosses $m\}$. $R^l_{0.5,1}(m)$ is the maximum correlation across m that represents the reliability score. We utilize it later in the Upsampling Refinement module. Fig. 1(b) shows all correlation pairs that cross certain points in $t = 0.5$. Since the max function does not facilitate the calculation of gradients, we use softmax for smoothing:

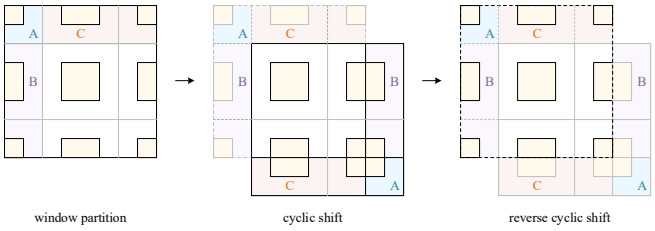

Fig. 3. Illustration of shifted window partitioning.

$$\begin{cases} F^l_{0.5\to1}(m) = \sum\limits_{(x_0,x_1)\in\mathbb{P}} \alpha_{(x_0,x_1)} \cdot (x_0 - x_1) * 0.5 \\ R^l_{0.5,1}(m) = \sum\limits_{(x_0,x_1)\in\mathbb{P}} \alpha_{(x_0,x_1)} \cdot C^l_{0,1}(x_0, x_1) \\ \alpha_{(x_0,x_1)} = \text{SoftMax}(C^l_{0,1}(x_0, x_1)) \end{cases} \tag{7}$$

We estimate the intermediate flows for each level separately. In the current module, we obtain $F^l_{0.5\to1}$ by taking the negative of $F^l_{0.5\to0}$ based on the linear motion.

Shifted Window. Computing all correlation pairs in $t = 0$ and $t = 1$ leads to a quadratic complexity $O(H \times W)$ about the features. Dividing the features into non-overlapping windows can effectively reduce the computational effort. However, calculating correlations based on windows leads to neglecting all cross-window motions, even if they are tiny. We refer to the shifted window proposed in Swin Transformer [25] to solve this problem, which introduces cross-window information while maintaining efficient computation of non-overlapping windows.

For the flow estimation, we find that more correlation pairs will pass closer to the center in Fig. 1(b). This finding can be generalized to the two-dimensional case. Therefore, we keep the center part of each window and discard the rest. Unlike Swin Transformer, which alternately calculates window attention and shifted window attention, we calculate the correlation for the four cases of the original, right shift, down shift, and right-down shift simultaneously and then put

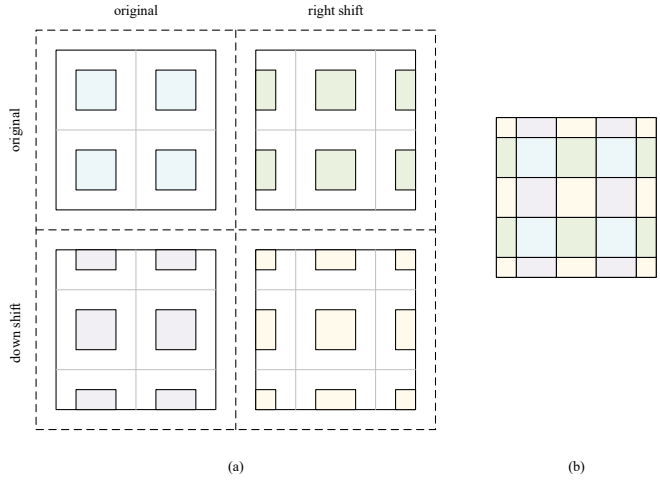

Fig. 4. The calculation process of the shifted window. (a) The parts that are preserved for each case. (b) The result of the splicing.

them together. Fig. 3 shows the shifted window partitioning for the right-down shift case. We calculate the correlation using a circular shift and then reverse the circular shift to obtain the result. The colored parts in Fig. 4(a) mark the preserved parts of the window in each case. Fig. 4(b) shows the result of putting together the preserved parts of each case. We set the window size for each level to $8, 16$, and 20 in our implementation.

Upsampling Refinement. We design this module for a more accurate upsampling of the flow. We consider the full-resolution flow a weighted combination of its nearby 3×3 flows at low resolution. As shown in Fig. 5, we start by sampling the flow to full resolution with a simple replication. We then use the unfold function provided by PyTorch [29] to extract the flows and reliability scores around each patch. Because the multiplication rate of sampling to full resolution is different for each level, we set the dilated rate to $2, 4, 8$, respectively. Fig. 5(b) illustrates the flows selected for a certain position when $l = 1$.

In the target frame, patches belonging to the same object are likelier to have the same flows. Therefore, we calculate the similarity scores of each patch and its nearby patches in the input frames. Likewise, we use the unfold function to extract the pixel values near each patch in the input frames. The dilated rate for each level is consistent with that mentioned earlier. Fig. 5(c) shows the patches selected in the input frame for a particular position. We use a convolutional network to obtain scores S_0^l and S_1^l from the processed frames. The resolution of the scores for all levels is the same as the input frame. ScoreNet is a small U-Net [30] with three downsampling layers, and we apply layernorm to each layer. Taking $F_{0.5\to1}^l$ as an example, we calculate the refined full-resolution flow $\widetilde{F}_{0.5\to1}^l$ by the following equation:

$$\begin{cases} \widetilde{F}_{0.5\to1}^l(x) = \sum_{b\in\mathbb{B}} \beta_{(x,b)} \cdot F_{0.5\to1}^l(b) \\ \beta_{(x,b)} = \text{SoftMax}(R_{0.5,1}^l(b) * S_{0.5}^l(x,b)) \end{cases} \tag{8}$$

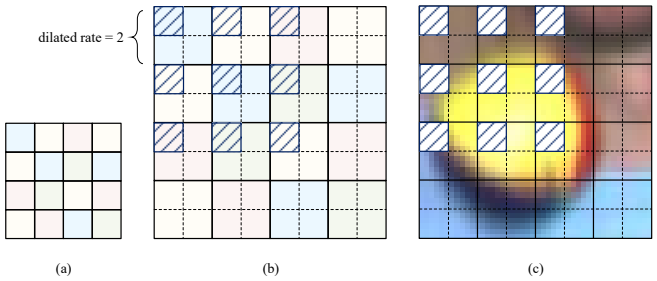

Fig. 5. Illustration of the 3×3 neighborhood at a particular position. Take level 1 as an example. (a) Low-resolution flows. (b) Full-resolution flows after a simple replication. (c) Full-resolution input frame.

where $\mathbb{B}$ is the set of neighbors of x. The reliability scores $R_{0.5,1}^l$ are obtained by following (7). We cannot get $S_{0.5}^l$ through ScoreNet since $I_{0.5}$ is unavailable. Hence, for each neighbor b of the patch, we take $S_1^l(x', b)$ as an approximation of $S_{0.5}^l(x, b)$, where x' is the corresponding position of x using the flow of b in I_1. Utilizing such a weighting design can refine the results of upsampling and thus improve the performance of the model. We demonstrate this in our ablation study.

C. Multi-scale Flow Fusion

We estimate multiple levels of flows in Sec. III-B. Since the RFE module calculates within a window, it cannot handle oversized motions. The range of flows that can be estimated for each level depends on the window size and the downsampling multiplier. In order to capture large motions without missing small ones, we designed the Multi-scale Flow Fusion (MFF) module. We calculate the range of flows that can be estimated for each position in each level and compare it with the actual estimated flow. If the flow estimated at the upper level exceeds the estimable flow range of the lower level, the former is selected. Conversely, the lower-level flow with more detail is selected. We use convolutional networks to generate the fusion masks for the flows.

The flow range in each position in the window is obviously different. At a certain level, the flows of an object may be out of range at some positions, resulting in a portion of the flows not being correctly estimated. It destroys the consistency of the flows of the same object and further causes the object to be partially missing in the intermediate frames. Hence, we use FusionNet to refine the fused flows. FusionNet is a small U-Net [30] with three downsampling layers.

D. Intermediate Frame Synthesis

Most of the objects in the intermediate frames will appear in both input frames. However, when object positions overlap, the occluded object does not appear in one of the frames. Therefore, it is essential to choose the appropriate frame for each position. We input the fused $F_{0\to1}$ and $F_{1\to0}$ into a convolutional network called MaskNet to obtain the occlusion mask. Then, we follow (3) to synthesize the preliminary intermediate frame.

Warping only the input frames to get intermediate frames ignores the abundant contextual information in the frames. We

TABLE I

QUANTITATIVE COMPARISONS ON THE UCF101 [31], VIMEO90K [32], HD [4] AND SNU-FILM [33] DATASETS. WE ALSO COMPARED THE PRE-TRAINING COMPONENTS AND MODEL PARAMETERS FOR EACH METHOD. FOR SIMPLICITY OF THE TABLE, WE ABBREVIATE SNU-FILM AS SF.

Methods	Pre-trained Components	UCF101 PSNR/SSIM	Vimeo90K PSNR/SSIM	HD PSNR	SF-Easy PSNR/SSIM	SF-Medium PSNR/SSIM	SF-Hard PSNR/SSIM	SF-Extreme PSNR/SSIM	Parameters (Million)
TOFlow [13]	SpyNet [34]	34.58/0.967	33.73/0.968	29.37	39.08/0.989	34.39/0.974	28.44/0.918	23.39/0.831	1.1
MEMC-Net [4]	FlowNetS [21]	35.01/**0.968**	34.40/0.974	31.39	-	-	-	-	70.3
SoftSplat [16]	PWC-Net [23]	**35.39**/ -	**36.10**/ -	-	-	-	-	-	7.7
DAIN [15]	PWC-Net [23] Megadepth [35]	34.99/**0.968**	34.71/**0.976**	31.64	39.73/**0.990**	35.46/**0.978**	30.17/**0.934**	**25.09**/**0.858**	24.0
CyclicGen [36]	HED [37]	35.11/**0.968**	32.09/0.949	-	37.72/0.984	32.47/0.955	26.95/0.887	22.70/0.808	3.0
CAIN [33]	×	34.91/**0.969**	34.65/0.973	31.77	39.89/**0.990**	35.61/**0.978**	29.90/0.929	24.78/0.851	42.8
SepConv [10]	×	34.78/0.967	33.79/0.970	30.87	39.41/**0.990**	34.97/0.976	29.36/0.925	24.31/0.845	21.6
AdaCoF [11]	×	34.90/**0.968**	34.47/0.973	31.43	39.80/**0.990**	35.05/0.975	29.46/0.924	24.31/0.844	22.9
EDSC [28]	×	35.13/**0.968**	34.84/0.975	31.59	**40.01**/**0.990**	35.37/**0.978**	29.59/0.926	24.39/0.843	8.9
BMBC [38]	×	35.15/**0.969**	35.01/**0.976**	-	39.90/**0.990**	35.31/0.977	29.33/0.927	23.92/0.843	11.0
MIFE_s	×	35.14/**0.968**	34.69/0.975	**31.86**	39.94/**0.990**	**35.68**/**0.978**	**30.29**/0.933	25.05/**0.856**	5.0
MIFE	×	**35.24**/**0.969**	**35.42**/**0.978**	**32.24**	**40.14**/**0.991**	**35.86**/**0.979**	**30.42**/**0.935**	**25.16**/**0.858**	5.9

extract the pyramid context features using Context Extractor, a convolutional network. Similarly, we warp these contextual features using the estimated intermediate flows. Finally, we input the original frames, the warped frames, the warped contextual features, the intermediate flows, and the occlusion mask into a small U-Net [30] to generate the residual frames. We add the residual frames to the intermediate frames obtained in the previous step to reduce the artifacts.

E. Implementation Details

Loss Function. We use Reconstruction Loss to supervising the reconstruction quality of intermediate frames. It is known that optimization based on the L_2 paradigm in most image synthesis tasks leads to blurred results [39], [40]. Thus, we use the Charbonnier function [41] to approximate the original L_1 loss. Reconstruction Loss is defined as the Charbonnier loss between the ground truth and the generated frame:

$$\mathcal{L}_{rec} = \rho(\widehat{I}_{0.5} - I^{GT}_{0.5}) \tag{9}$$

where $\rho(x) = (x^2 + \epsilon^2)^{1/2}$ and ϵ is set to 0.001. To predict motions of different sizes, we supervise the flows at each level using Multi-scale Loss:

$$\mathcal{L}_{ms} = \sum_{l=1}^{3} \rho(\widetilde{I}^{l}_{0.5} - I^{GT}_{0.5}) \tag{10}$$

where $\widetilde{I}^{l}_{0.5}$ is the frame generated by the flows of level l following (3). Finally, we combine the above losses linearly to obtain our loss function:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{ms}\mathcal{L}_{ms} \tag{11}$$

where we set $\lambda_{ms} = 0.1$ in the experiments.

Training Dataset. We train MIFE from scratch using the Vimeo90K dataset [32]. It contains 51,312 triplets of 256 × 448 video frames, and we randomly crop the frames to 256 × 256 during training. Moreover, we augment the training data by randomly flipping the frames horizontally and vertically as well as reversing the temporal order.

Training Strategy. MIFE is optimized by AdaMax [42] with $\beta_1 = 0.9$, $\beta_2 = 0.999$. We use a batch size of 16 and train the network for 80 epochs. The learning rate is initially 1×10^{-3} and is gradually reduced to 1×10^{-5} using cosine annealing during the training process.

IV. EXPERIMENTS

A. Evaluation Datasets and Metrics

UCF101. UCF101 [31] is an action recognition dataset containing various human actions. We evaluate 379 triplets with a resolution of 256×256 selected by DVF.

Vimeo90K. The Vimeo90K test dateset [32] is widely used to evaluate the performance of video frame interpolation methods. It contains 3782 triples with a resolution of 448×256.

HD. The HD dataset [4] is a collection of high-resolution videos from the Xiph website and the Sintel dataset [43]. It includes four 1920×1080, three 1280×720 and four 1280×544 videos. Consistent with the original paper, we evaluate the first 100 frames of each video.

SNU-FILM. The SNU-FILM dataset [33] is based on high frame rate videos, including videos from YouTube and the GOPRO dataset [44]. The maximum resolution of this dataset is 1280×720. It contains four subsets: easy, medium, hard and extreme, each consisting of 310 triples. Subsets of different difficulties have different degrees of motion.

Metrics. We evaluate the performance of each method by calculating its Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) [45]. Larger PSNR and SSIM mean a higher quality of the interpolated frames. For all tables in this section, numbers in red indicate the best performance, and numbers in blue indicate the second best performance.

B. Comparisons with Previous Methods

We compare the proposed approach with several competitive methods: TOFlow [13], MEMC-Net [4], SoftSplat [16], DAIN

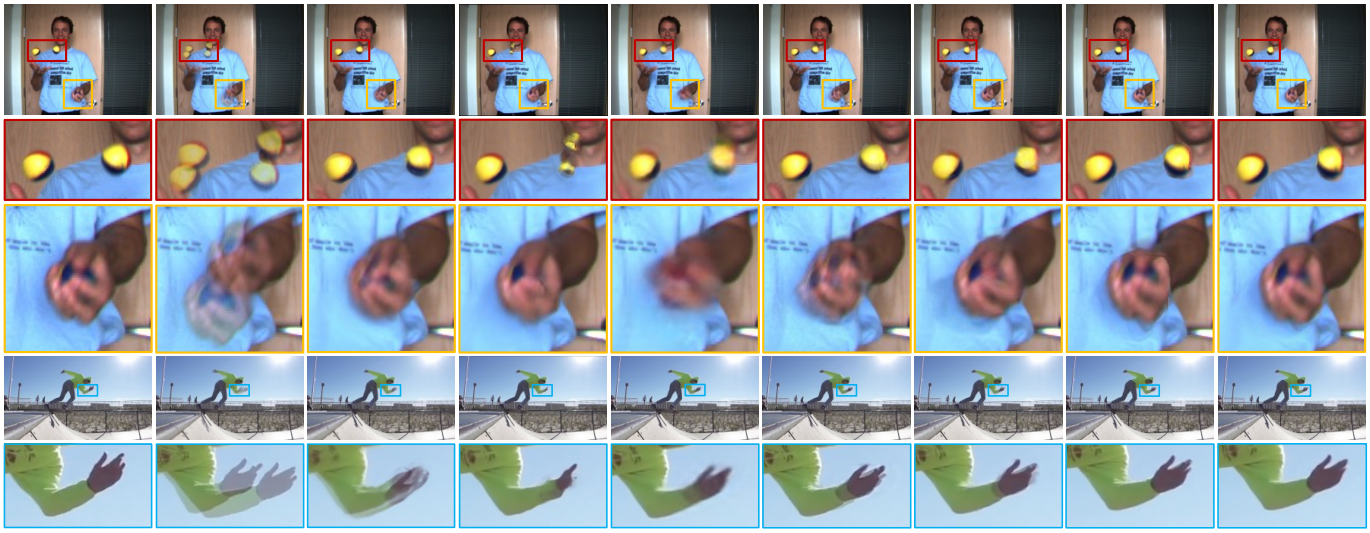

Fig. 6. Visual comparisons of large motions. The first three rows of images are from the Middlebury dataset [20]. The last two rows of images are from the SNU-FILM (Extreme) [33]. We cut out the objects and zoom in on them.

[15], CyclicGen [36], CAIN [33], SepConv [10], AdaCoF [11], EDSC [28], BMBC [38]. All results are reported by the original authors or obtained by executing the existing source code. Both the full version of the proposed method and the simplified version that only warps the input frames are involved in the comparison, respectively denoted as MIFE and MIFE_s.

Table I compares the average PSNR and SSIM scores for the UCF101, Vimeo90K, HD, and SNU-FILM datasets. On the UCF101 and Vimeo90K datasets, our method outperforms most methods that use only two consecutive frames as input. Unlike other flow-based methods, our method does not require pre-trained components. In addition, our parameters are significantly less than those of other competing methods such as DAIN and CAIN. On the HD and SNU-FILM datasets, even MIFE_s outperforms most competitors. HD and SNU-FILM (Hard, Extreme) have higher resolution and more complex motion than UCF101 and Vimeo90K. The advantages of our method become even more apparent with these challenging datasets.

Video frame interpolation aims to obtain frames with higher visual performance. PSNR and SSIM cannot ideally compare the various methods. We demonstrate the visual comparisons of large motions in Fig. 6. It can be seen that the methods involved in the comparison all exhibit varying degrees of overlap, blurring, or missing parts. In contrast, our method generates more precise and more realistic results. Our method handles fast motion between consecutive frames more efficiently than competitors.

C. Ablation Study

We analyze the contributions of the three components of our approach: Upsampling Refinement, Multi-scale Flow Fusion, and Multi-scale Loss. We perform the ablation experiments on a simplified version, namely MIFE_s.

Upsampling Refinement. To demonstrate the effectiveness of the Upsampling Refinement module, we compare it with several different upsampling methods. The first part of Table II shows the performance of several upsampling methods. The last row of the table shows the proposed method, which uses $\mathrm{SoftMax}(R*S)$ as the weight for upsampling. As can be seen, bilinear upsampling is the least effective of all methods since it does not consider the wholeness of the same object. Our method outperforms methods that use similarity or reliability scores alone because it takes full advantage of both. It selects similar and more plausible flows for each patch to improve the quality of the interpolated frames.

Multi-scale Flow Fusion. We analyze the contribution of the Multi-scale Flow Fusion module in the second part of Table II. The model with only a single level is far inferior to the model that fuses all levels of flows, regardless of which level is retained. Level 2 obtained the best results among all single levels. Level 1 is better at handling datasets with small motions, while level 3 is better at handling datasets with large motions. Our method fuses multiple levels of flows and can therefore handle datasets with different motion sizes.

The method without flow range performs well on simple datasets and even slightly outperforms ours. However, it is far inferior to the proposed method on datasets with large motion between frames like HD and SNU-FILM (Hard, Extreme). Taking the first set of frames in Fig.6 as an example, Fig.7 shows the visualization of fusion masks for the flows between levels. The yellow part indicates the selection of upper-level flows with larger ranges, and the purple part indicates the selection of more refined lower-level flows. It can be seen that the yellow part in Fig.7 contains the balls and the hands,

TABLE II

ABLATION STUDY ON UPSAMPLING REFINEMENT, MULTI-SCALE FLOW FUSION, AND MULTI-SCALE LOSS. WE REPORT THE PSNR AND SSIM OF THE MODELS WITH DIFFERENT SETTINGS ON THE UCF101 [31], VIMEO90K [32], HD [4] AND SNU-FILM [33] DATASETS.

Experiments	Settings	UCF101	Vimeo90K	HD	SF-Easy	SF-Medium	SF-Hard	SF-Extreme
		PSNR/SSIM	PSNR/SSIM	PSNR	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
Upsampling Refinement	bilinear	**35.11**/**0.9682**	34.47/0.9734	31.58	39.72/0.9898	35.54/0.9776	30.06/0.9316	24.67/0.8496
	SoftMax(S)	**35.11**/**0.9680**	**34.63**/**0.9743**	**31.82**	**39.87**/**0.9900**	**35.64**/**0.9779**	30.22/0.9328	25.00/0.8551
	SoftMax(R)	**35.14**/**0.9682**	34.55/0.9740	31.81	39.77/0.9898	35.62/0.9778	**30.27**/**0.9334**	**25.03**/**0.8556**
	SoftMax($R * S$)	**35.14**/**0.9682**	**34.69**/**0.9747**	**31.86**	**39.94**/**0.9901**	**35.68**/**0.9780**	**30.29**/**0.9332**	**25.05**/**0.8561**
Multi-scale Flow Fusion	$l = 1$	34.86/0/9663	33.02/0.9595	29.01	39.06/0.9886	33.58/0.9652	27.39/0.8966	22.88/0.8130
	$l = 2$	**34.92**/**0.9668**	**33.93**/**0.9701**	**31.17**	**39.39**/**0.9892**	**35.17**/**0.9764**	**29.60**/0.9270	24.16/0.8388
	$l = 3$	34.70/0.9659	33.45/0.9662	31.10	38.86/0.9883	34.75/0.9748	29.50/**0.9274**	**24.46**/**0.8461**
	w/ all levels	**35.14**/**0.9682**	**34.69**/**0.9747**	**31.86**	**39.94**/**0.9901**	**35.68**/**0.9780**	**30.29**/**0.9332**	**25.05**/**0.8561**
	w/o range	**35.14**/**0.9682**	**34.72**/**0.9747**	**31.80**	**39.94**/**0.9901**	**35.70**/**0.9780**	**30.19**/**0.9324**	**24.87**/**0.8525**
	w/o FusionNet	**34.96**/**0.9669**	34.11/**0.9714**	31.41	**39.58**/**0.9895**	35.24/**0.9766**	29.81/0.9293	24.63/0.8483
	w/ all components	**35.14**/**0.9682**	**34.69**/**0.9747**	**31.86**	**39.94**/**0.9901**	**35.68**/**0.9780**	**30.29**/**0.9332**	**25.05**/**0.8561**
Multi-scale Loss	w/o $\mathcal{L}_{ms}$	**35.06**/**0.9680**	**34.12**/**0.9700**	**30.98**	**39.60**/**0.9896**	**34.86**/**0.9746**	**29.00**/**0.9186**	**23.98**/**0.8340**
	w/ $\mathcal{L}_{ms}$	**35.14**/**0.9682**	**34.69**/**0.9747**	**31.86**	**39.94**/**0.9901**	**35.68**/**0.9780**	**30.29**/**0.9332**	**25.05**/**0.8561**

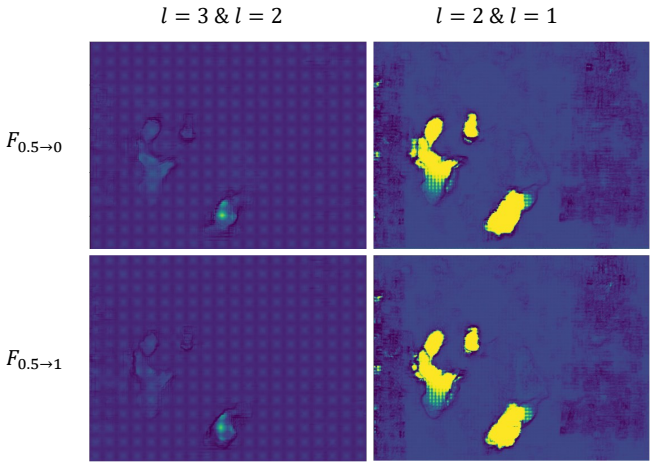

Fig. 7. Visualization of fusion masks for the flows between levels.

which is consistent with the area of large movements between frames. Our method can accurately capture large movements between consecutive frames without losing other details.

FusionNet contributes significantly to improving intermediate flows since the method without FusionNet presents poor results on all datasets. MFF achieves a balance between the fineness and the estimable flow range. It allows our method to perform well on evaluation datasets of varying difficulty.

Multi-scale Loss. Multi-scale Loss supervises the flows of each level. The third part of Table II shows the contribution of Multi-scale Loss. It dramatically improves the quality of the interpolated frames. In particular, the method without $\mathcal{L}_{ms}$ performs worse on more complex datasets. Multi-scale Loss assists the MFF module well and makes it more effective.

V. CONCLUSIONS

This paper proposes a new flow-based video frame interpolation method called Multi-scale Intermediate Flow Estimation (MIFE). In order to capture large motions without missing small motions, our method fuses multiple levels of refined flows. Unlike other flow-based methods, our approach does not require pre-trained components, thus avoiding the difficulty of manually labeling optical flows. With the Multi-scale Flow Fusion module adaptively selecting the desired flows, MIFE can efficiently process video with different resolutions and ranges of motion. Experiments show that our method can handle fast motions between consecutive frames more efficiently than other state-of-the-art methods.

ACKNOWLEDGMENT

This work is partially supported by National Key R&D program of China (2021ZD0113203), and National Science Foundation of China (61976115,61732006).

REFERENCES

[1] R. Castagno, P. Haavisto, and G. Ramponi, "A method for motion adaptive frame rate up-conversion," *IEEE Transactions on circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 436–446, 1996.

[2] H. Choi and I. V. Bajić, "Deep frame prediction for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1843–1855, 2019.

[3] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. 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*, 2018, pp. 9000–9008.

[4] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang, "Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 3, pp. 933–948, 2019.

[5] W. Shen, W. Bao, G. Zhai, L. Chen, X. Min, and Z. Gao, "Blurry video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5114–5123.

[6] X. Xiang, Y. Tian, Y. Zhang, Y. Fu, J. P. Allebach, and C. Xu, "Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3370–3379.

[7] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1410–1418.

[8] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers, "Phasenet for video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 498–507.

[9] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[10] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[11] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee, "Adacof: Adaptive collaboration of flows for video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5316–5325.

[12] X. Cheng and Z. Chen, "Video frame interpolation via deformable separable convolution," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 607–10 614.

[13] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[14] S. Gui, C. Wang, Q. Chen, and D. Tao, "Featureflow: Robust video interpolation via structure-to-texture generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 004–14 013.

[15] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3703–3712.

[16] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5437–5446.

[17] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1701–1710.

[18] M. Werlberger, T. Pock, M. Unger, and H. Bischof, "Optical flow guided tv-l 1 video interpolation and restoration," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2011, pp. 273–286.

[19] Z. Yu, H. Li, Z. Wang, Z. Hu, and C. W. Chen, "Multi-level video frame interpolation: Exploiting the interaction among different levels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1235–1248, 2013.

[20] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[21] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[22] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[23] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[24] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*. Springer, 2020, pp. 402–419.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[26] G. Wolberg, H. Sueyllam, M. Ismail, and K. Ahmed, "One-dimensional resampling with inverse and forward mapping functions," *Journal of Graphics Tools*, vol. 5, no. 3, pp. 11–33, 2000.

[27] F. A. Reda, D. Sun, A. Dundar, M. Shoeybi, G. Liu, K. J. Shih, A. Tao, J. Kautz, and B. Catanzaro, "Unsupervised video interpolation using cycle consistency," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 892–900.

[28] X. Cheng and Z. Chen, "Multiple video frame interpolation via enhanced deformable separable convolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[31] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[32] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[33] M. Choi, H. Kim, B. Han, N. Xu, and K. M. Lee, "Channel attention is all you need for video frame interpolation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 663–10 671.

[34] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[35] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *Advances in neural information processing systems*, vol. 29, 2016.

[36] Y.-L. Liu, Y.-T. Liao, Y.-Y. Lin, and Y.-Y. Chuang, "Deep video frame interpolation using cyclic frame generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8794–8802.

[37] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[38] J. Park, K. Ko, C. Lee, and C.-S. Kim, "Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation," in *European Conference on Computer Vision*. Springer, 2020, pp. 109–125.

[39] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *European Conference on Computer Vision*. Springer, 2016, pp. 434–450.

[40] F. A. Reda, G. Liu, K. J. Shih, R. Kirby, J. Barker, D. Tarjan, A. Tao, and B. Catanzaro, "Sdc-net: Video prediction using spatially-displaced convolution," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 718–733.

[41] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of 1st International Conference on Image Processing*, vol. 2. IEEE, 1994, pp. 168–172.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European conference on computer vision*. Springer, 2012, pp. 611–625.

[44] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3883–3891.

[45] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.